HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

DIPLOMARBEIT

Herr Ing.
Wolfgang Goßner

Eignung agiler Methoden für die Softwareentwicklung bei ERP-Implementierungspartnern

Mittweida, 2016

Fakultät
Angewandte Computer -und Biowissenschaften

DIPLOMARBEIT

Eignung agiler Methoden für die Softwareentwicklung bei ERP-Implementierungspartnern

Autor:
Herr Ing.
Wolfgang Goßner

Studiengang:
Informationstechnik

Seminargruppe:
KI09w2IA

Erstprüfer:
Prof. Dr. Ing. Volker Delport

Zweitprüfer:
Dipl. Wirtschaftsing. (FH) Günther Förster

Einreichung:
Mittweida, 31. Juli, 2016

Verteidigung/Bewertung:
Mittweida, 2016

Faculty Applied Computer Sciences
& Bioscience

DIPLOMA THESIS

Suitabily of agile methods for softwaredevelopment for ERP-implementationpartners

author:
Mr. Ing.
Wolfgang Goßner

course of studies:
informationtechnologie

seminar group:
KI09w2IA

first examiner:
Prof. Dr. Ing. Volker Delport

second examiner:
Dipl. Wirtschaftsing. (FH) Günther Förster

submission:
Mittweida, July 31st 2016

defence/ evaluation:
Mittweida, 2016

Bibliografische Beschreibung:

Goßner, Wolfgang:

Eignung agiler Methoden für die Softwareentwicklung bei ERP-Implementierungspartnern. - 2016. – 3 Seiten Verzeichnisse, Inhalt 65 Seiten, 15 Abbildungen, Hochschule Mittweida, Fakultät Angewandte Computer -und Biowissenschaften

Diplomarbeit, 2016

Referat:

In der Softwareentwicklung gewinnen agile Methoden zunehmend an Bedeutung. ERP-Implementierungspartner sind in der Regel keine Entwickler neuer Standardanwendungen, sondern erweitern oder ergänzen vom Hersteller gelieferte Standardsoftware.

Die Diplomarbeit soll zeigen, ob agile Methoden für die Softwareentwicklung bei ERP-Implementierungspartnern aufgrund der besonderen Anforderungen geeignet sind.

Inhalt

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Terna ist Anbieter von ERP-Software (Enterprise Ressource Planning) und Implementierungspartner für Infor M3, Microsoft Dynamics AX und Microsoft Navision. Weiteres werden Produkte im Umfeld von BI (Business Intelligence) und CRM (Customer Relationship Management) vertrieben und bei Kunden eingeführt. Der Hauptsitz des Unternehmens ist in Innsbruck in Tirol (Österreich).

Ein Team von 150 Programmierern, Beratern und Technikern in Österreich, Deutschland, in der Schweiz und in Indien konzipiert und programmiert Erweiterungen zu der vom Hersteller gelieferten Standardsoftware, sowie eigenständige Anwendungen, die über Schnittstellen mit den ERP-Systemen kommunizieren.

Das Geschäft hat sich in den letzten Jahren nicht so schnell wie in anderen IT-Bereichen verändert. Es sind aber vor allem der große Wettbewerb und die gestiegenen Kundenanforderungen, die ein Umdenken und eine Anpassung der Methoden notwendig machen. Die Kunden wollen heute viel schneller und zu geringeren Kosten Lösungen geliefert und Kostenobergrenzen per Vertrag zugesichert bekommen.

Nachdem mich der Entwicklungsleiter auf Scrum aufmerksam gemacht und mir ein kleines Buch in Form eines Wirtschaftsromans empfohlen hatte, war in mir das Interesse an agilen Vorgehensmodellen geweckt worden.

Scrum wurde damals aus verschiedensten Gründen bei Terna nicht eingeführt, aber die Erkenntnis, dass es Veränderungen bedarf war in der Zwischenzeit bereits weit gereift. Terna hat alle notwendigen Voraussetzungen geschaffen und in vielen Workshops und Diskussionen eine für das Unternehmen passende, agile Implementierungsmethode beschrieben und als vorläufigen Standard festgelegt. Es wurden auch schon erste Projekte erfolgreich agil abgewickelt und fertiggestellt.

1.2 **Arbeit**

Im Kapitel 2 werden das Geschäftsumfeld und die besonderen Anforderungen von ERP-Implementierungspartnern an die Softwareentwicklung dargestellt.

Kapitel 3 beschäftigt sich mit den klassischen Methoden, vergleicht sie mit dem früher bei Terna verwendeten Vorgehensmodell und bewertet dieses aus Sicht der Kunden und der Geschäftsleitung.

Eine Beschreibung ausgewählter agiler Methoden und der Anforderungen von Terna erfolgen in Kapitel 4. Untersuchungen hinsichtlich deren Eignung und die Ableitung einer für Terna idealen Methode schließen das Kapitel ab.

Relevante Auszüge aus der bei Terna eingeführten Methode werden im Kapitel 5 vorgestellt.

Kapitel 6 fasst bisherige Erfahrungen und Erkenntnisse zusammen und versucht die Frage zu beantworten, ob agile Methoden für die Softwareentwicklung bei ERP-Implementierungspartnern geeignet sind.

2 ERP-Implementierungspartner

In diesem Kapitel werden die Besonderheiten von ERP-Software und die besonderen Anforderungen an die Entwicklung beschrieben, und was man unter ERP-Implementierungspartner versteht.

2.1 Software

"Ein Enterprise-Ressource-Planning-System oder kurz ERP-System dient der funktionsbereichsübergreifenden Unterstützung sämtlicher in einem Unternehmen ablaufenden Geschäftsprozesse. Entsprechend enthält es Module für die Bereiche Beschaffung/Materialwirtschaft, Produktion, Vertrieb, Forschung und Entwicklung, Anlagenwirtschaft, Personalwesen, Finanz- und Rechnungswesen, Controlling usw., die über eine gemeinsame Datenbasis miteinander verbunden sind. Durch die unternehmensweite Konsolidierung der Daten ist eine Unterstützung der Planung über sämtliche Unternehmensebenen hinweg (von der Konzernebene über verschiedene Werke, Sparten und Abteilungen bis hin zu einzelnen Lagerorten) möglich.“ [Wirtschaftslexikon 2016-1]

"ERP II-Systeme beinhalten eine Erweiterung klassischer ERP-Systeme um Funktionen zur Unterstützung unternehmensübergreifender Prozesse. Der Fokus liegt dabei auf einer durchgängigen Prozessunterstützung und dem zwischenbetrieblichen Informationsaustausch durch standardisierte Komponenten und internetbasierte Schnittstellen (sog. Service-orientierte Architekturen, abgekürzt SOA)." [Wirtschaftslexikon 2016-2]

Bei der Programmierung muss der großen Anzahl Transaktionen besondere Beachtung geschenkt werden. Aufgrund der Datenmengen müssen mehrfache Haltung selbiger vermieden werden, weshalb relationale, große Datenbanken zum Einsatz kommen. Die Anwendungen können durch Konfiguration und Parametrierung an die Geschäftsprozesse angepasst werden. Hier spricht man noch nicht von Programmierung. Es ist möglich unzählige Varianten für Prozessabläufe und somit Abläufe in der Software einzustellen, die trotz weiter Verbreitung der Software, nicht alle getestet sind.

Die Nutzung der Software ist ohne Schulung und Beratung, sowie einem entsprechenden Setup in der Regel nicht möglich.

Der Hersteller stellt nur den Standard zur Verfügung. Neue Releases und Fehlerkorrekturen können nur bei Vorhandensein von Wartungsverträgen bezogen werden. Kundenspezifische Anpassungen werden vom Hersteller nur in Ausnahmefällen gewartet und zwar nur dann, wenn der Hersteller selbst oder ein entsprechend zertifizierter Partner die Programmierungen vorgenommen hat und entsprechende Zertifizierungen vorliegen.

Die wichtigsten Anbieter auf dem europäischen Markt sind:

- SAP mit R3
- Oracle mit People Soft
- Microsoft mit Dynamics AX und Navision
- Infor mit M3 und LN
- proALPHA
- APplus
- abas

Die Software wird heute noch häufig On-Premise (in den eigenen Räumlichkeiten) betrieben. Die meisten Hersteller bieten aber auch schon den Betrieb der Software in einer Cloud[1] an.

Terna verkauft und implementiert die ERP-Produkte Infor M3, Microsoft Dynamics AX und in der Firmengruppe auch Microsoft Navision für Kunden aus dem Mittelstand.

2.2 Implementierungspartner

Bei ERP-Implementierungspartnern handelt es sich in der Regel um Software -und Beratungsunternehmen, die Partnerverträge mit Herstellern von ERP-Software eingegangen sind. Es gibt unterschiedliche Arten von Verträgen, die unter anderem regeln, welche Dienstleistungen der Partner in Zusammenhang mit den Produkten des Herstellers liefern darf oder liefern muss.

Terna und auch andere Partner erbringen im Rahmen dieser Partnerverträge folgende Leistungen:

- Verkauf von Software
- Installation von Software beim Kunden oder in einem Rechenzentrum
- Konfiguration und Parametrierung der Anwendung[2]
- Schulung in Nutzung und Betrieb der Software
- Lieferung kundenspezifischer Anpassungen und Erweiterungen
- Wartung der Software

Zur Erbringung dieser Dienstleistungen beschäftigt Terna Berater, Programmierer, Systemtechniker und Projektleiter.

[1] Cloud Computing, Rechnernetzwerk, onlinebasierte Speicher- und Serverdienste

[2] Teil der Software mit Businesslogik

2.3 **besondere Anforderungen**

In diesem Kapitel werden die besonderen Anforderungen im ERP-Umfeld beschrieben.

Hardware

Moderne ERP-Software ist plattformunabhängig und objektorientiert. Hardware wird heutzutage nicht mehr gemeinsam mit Software verkauft. Hersteller von Großrechnern, wie IBM, gehen davon aus, dass das Geschäft künftig nicht mehr mit Maschinen beim Kunden vor Ort, sondern mit in Rechenzentren gemacht werden wird. Die Softwarehersteller und auch die Partner wollen, damit sie auch am Hardwaregeschäft teilhaben können, Software in Clouds oder Rechenzentren (Private Cloud) betreiben. Die besonderen Anforderungen in der Cloud sind eigentlich eher softwareseitig und aus Sicht der Datensicherheit zu sehen. Die Hersteller möchten aus technischen Gründen, nur mehr Standardsysteme und aktuelle Versionen und Releases in der Cloud betreiben. In der Regel haben die Kunden aber spezielle Anforderungen, die nur zu einem gewissen Teil über Konfiguration und Parametrierung abgebildet werden können. Releasewechsel sind im ERP-Umfeld aufgrund des großen betriebswirtschaftlichen Risikos, mit einem erheblichen Testaufwand verbunden. Die Entwicklungen sind an diese Stelle heute noch schwer abschätzbar.

Umfeld Software

Hersteller liefern die Software und stellen neue Releases und Fehlerkorrekturen bereit. Partner arbeiten meist nicht an der Entwicklung des Herstellerstandards mit, haben aber häufig eigene Erweiterungen zum Standard programmiert. Im Falle von Microsoft Dynamics AX gibt es Partner, die sich auf bestimmte Branchen konzentriert haben und dafür entsprechende Lösungen anbieten. Terna bietet ebenfalls Branchenlösungen an, z.B. für den Apparate -und Gerätebau, sowie für Chemie und Pharma. In der Regel werden aber Erweiterungen und Zusatzanwendungen auf Basis von Kundenwünschen nach Konzeption durch den Berater programmiert.

Implementierungspartner, wie auch Terna, bieten häufig auch Zusatzsoftware an, die über Schnittstellen mit dem ERP-System zusammenarbeiten. Diese Softwareprodukte müssen auch auf Basis eines Standards weiterentwickelt und gewartet werden. In diesem Falle spricht man dann auch von Standardentwicklung, häufig aber in kleinerem Umfeld.

Risiko

Fehler in der Programmierung gefährden zwar keine Menschenleben, können aber großen betriebswirtschaftlichen Schaden, z.B. wegen verlorener Umsätze, Mehraufwände, Fehlfakturierungen, usw. zur Folge haben. Die Anforderungen an Qualität und Tests zur Reduktion des Risikos sind speziell bei größeren Anwendungen sehr hoch.

Kaufmännisches Umfeld

In früheren Jahren wurden häufig Projektbudgets gemeinsam vereinbart, die Abrechnung ist dann aber nach Aufwand (Time & Material) erfolgt. In letzter Zeit verlangen die Kunden aber Festpreise, die auf Basis von Designdokumenten oder Pflichtenheften vereinbart werden. Änderungen werden dann in der Folge über Changerequests abgewickelt. Die Mehrkosten müssen dann vom Kunden getragen werden. Dies führt immer wieder zu großen Diskussionen über Formulierungen und deren Auslegung in den Dokumenten und letztlich zu unzufriedenen Kunden.
Bei der Vereinbarung der Budgets und der Preise ist eine Orientierung an Mitbewerben notwendig. Häufig sind die Anforderungen im Detail zu diesem Zeitpunkt jedoch noch gar nicht bekannt.

Die Kunden wollen wegen der einmaligen und laufenden Kosten und in jüngster Zeit auch wegen des Betriebes in der Cloud, keine individuellen Programmierungen beauftragen.

Projektmanagement

Terminverschiebungen in Teilbereichen können Auswirkungen auf das Gesamtprojekt haben. Die Folge sind zusätzliche Kosten bei Kunde und Lieferant und verärgerte Kunden.

Vorgehensmodelle

Es wird häufig noch nach dem Wasserfall oder V-Modell gearbeitet, teilweise auch nach dem Spiralmodell. Viele Unternehmen, z.B. aus der Pharmaindustrie wollen und müssen Projekte nach diesen und ähnlichen Vorgehensmodellen abwickeln. Es gibt aber auch viele Kunden und Interessenten, die andere, agilere Modelle bevorzugen und es wünschen, dass z.B. regelmäßig Zwischenlösungen oder Prototypen präsentiert werden, die auch bewertet werden können.

Die meisten Hersteller stellen auch Methoden, Datenbanken, Dokumentvorlagen und Werkzeuge zur Verfügung, um Projekte nach einem vom Hersteller gewünschten Vorgehensmodell abzuwickeln.

3 Klassische Methoden

In diesem Kapitel werden die wichtigsten klassischen Methoden und die bisher von Terna verwendete Methode beschrieben und bewertet.

3.1 Klassische Methoden

Bei der NATO-Konferenz 1968 in Garmisch-Partenkirchen wurde als Reaktion auf die Softwarekrise eine ingenieurmäßige Vorgehensweise bei der Entwicklung von Software und Systemen vorgegeben. Der Begriff Software Engineering wurde geprägt, und es wurden viele Verfahren und Methoden erfunden und umfangreiche Anweisungen und Regelungen erstellt. [Hruschka, Peter; Rupp, Chris; Starke, Gernot 2009]

Boehm und Turner [Boehm und Turner 2004] sprechen auch von plangetriebener Entwicklung, basierend auf Konzepten, die eher aus dem Ingenieursbereich kommen. Man versuchte die Software zu „disziplinieren“ [Boehm und Turner 2004]. Zu diesem Zwecke wurde eine Reihe von Richtlinien erlassen. Federführend dabei war in Amerika die Verteidigungsindustrie, so sind einige der Vorschriften im MIL-STD-1521, DoD-STD-2167, MIL-STD-498 zu finden, aber auch in der ISO 9000. Es gibt auch Softwarehäuser, wie IBM, Hitachi und Siemens, die eigene Standards festgelegt haben. [Boehm und Turner 2004]

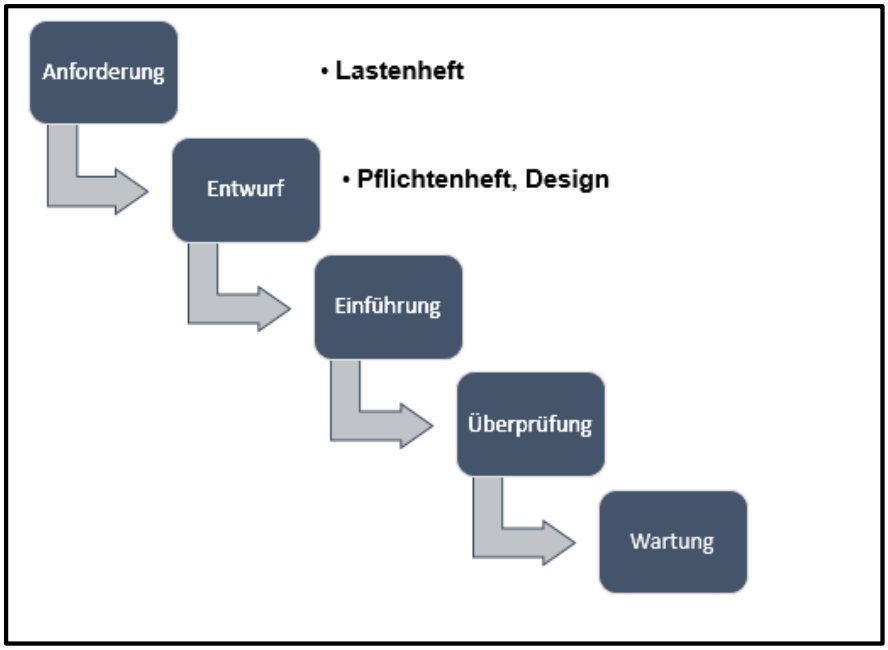
Charakteristik
Klassische bzw. plangetriebene Methoden haben einen ingenieurmäßigen Ansatz, in dem man an spezifischen Prozessen in den verschiedenen Entstehungsphasen festhält. Es gibt den Bedarf vollständiger Dokumentation auf jeder Stufe, damit eine sorgfältige Verifikation erreicht wird.

Im Nachfolgenden wird eine Auswahl bekannter „klassischer“ Methoden beschrieben.

3.1.1 Das Wasserfallmodell

Das Wasserfallmodell ist ein lineares (nicht iteratives) Vorgehensmodell, das in Phasen organsiert ist. Am Ende einer jeden Phase werden Dokumente erzeugt, die Bedingung für den Abschluss der Phase sind und die die Basis für die Tätigkeiten der nächsten Phase bilden. Üblicherweise gibt es fünf Phasen, aber auch Modelle mit sechs und mehr Phasen kommen in der Praxis vor.

Abb. 1: Wasserfallmodell



Das Wasserfallmodell wurde von Winston W. Royce beschrieben, der aber schon 1970 erkannt hat, dass das Modell verbesserungswürdig ist. Unter anderem sollte ein Prototypenansatz zur Anwendung kommen und der Kunde sollte in Form von Reviews eingebunden

Im Laufe der Zeit wurden auch Erweiterungen entwickelt mit denen es im Fehlerfalle möglich ist, schrittweise die Kaskade wieder nach oben zu laufen.

Änderungen sind in diesem Modell eigentlich gar nicht vorgesehen, was speziell bei Projekten mit langer Durchlaufzeit stets problematisch ist. Zusatzkosten und Ärger innerhalb aller Beteiligten sind die Folge. Das reine Wasserfallmodell wird in der Praxis kaum angewendet. In der Regel gibt es Abwandlungen, die Iterationen ermöglichen und Änderungen werden auch nach Abschluss der Phasen erlaubt. Man darf aber einen wesentlichen Vorteil der Methode nicht außer Acht lassen: Die Abfolge ist, wenn auch in der Praxis kaum anwendbar, im Grunde genommen logisch, wiederholbar, standardisiert und vor allem für Jedermann leicht verständlich.

3.1.2 Code and Fix

Dabei handelt es sich um ein eher unstrukturiertes Vorgehen ohne Vorhandensein einer Spezifikation bei dem so lange codiert und korrigiert wird, bis das gewünschte Verhalten erreicht ist. [Ludewig und Lichter 2010].

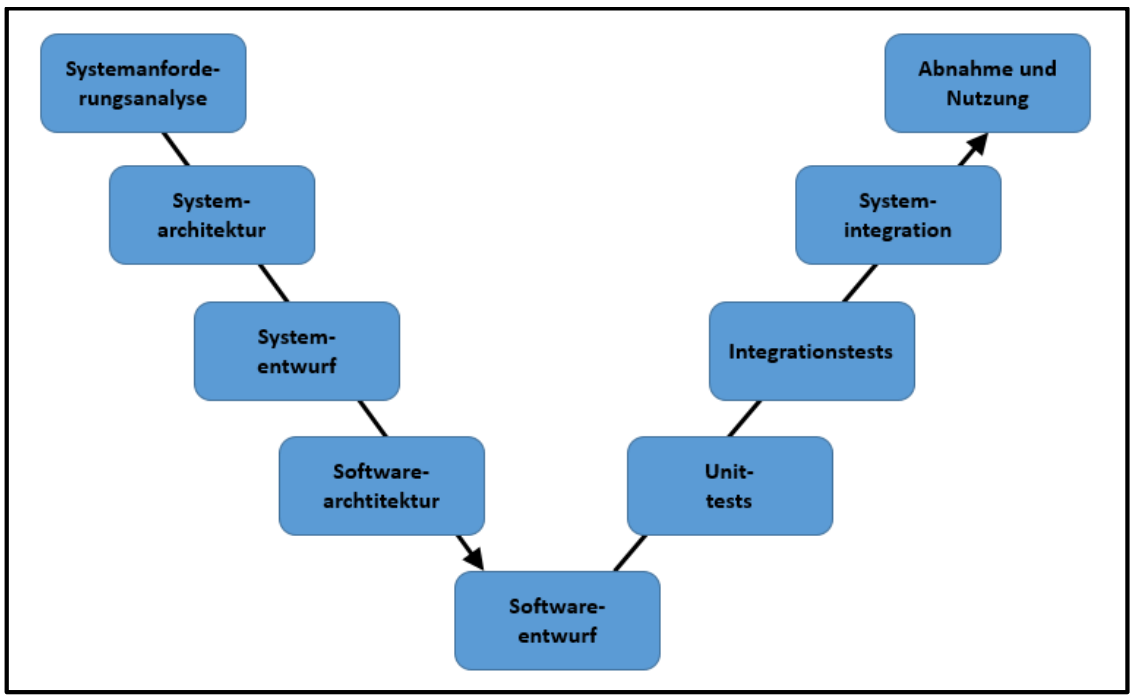
Dieses Vorgehensmodell wird meist unbewusst dann eingesetzt, wenn man ohne Spezifikation und soliden Entwurf Programmcode eintippt [Ludewig und Lichter 2010].

3.1.3 Das V-Modell

Das V-Modell ist vom Wasserfallmodell abgeleitet und wurde von Barry Boehm 1979 beschrieben. Die Phasen sind jeweils mit Qualitätssicherung und Tests verbunden. Öffentliche deutsche Auftraggeber schreiben im Sinne einer Vereinheitlichung der Standards bei der Vergabe von Entwicklungsaufträgen die Anwendung des V-Modells seit 1992 für den Verteidigungsbereich vor, seit 1996 findet es auch im zivilen Bereich Anwendung. [Ludewig und Lichter 2010]

Weiterentwicklungen gab es 1997 mit dem V-Modell 97, welches auch Hardwareentwicklung, sowie inkrementelle Vorgehen unterstützt. Im Jahre 2004 wurde die Version V-Modell XT vorgestellt. Die Erweiterung XT bedeutet extreme Tailoring und soll die Anpassbarkeit zum Ziel haben. [Ludewig und Lichter 2010]

Abb. 2: V- Modell



Merkmale des V- Modells XT nach Ludewig und Lichter [Ludewig und Lichter 2010]:

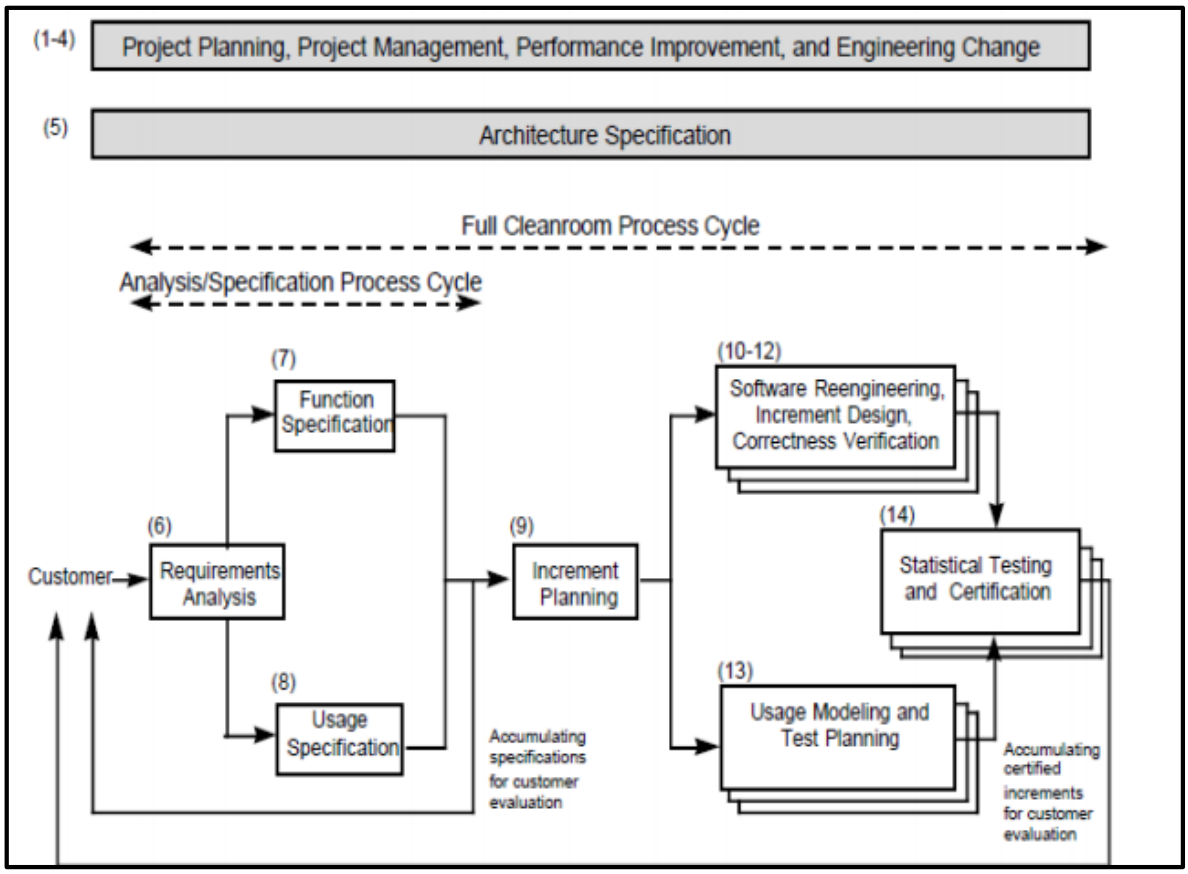
- Es erfolgt eine Aufteilung in Phasen, an dessen Ende jeweils ein Meilenstein als Entscheidungspunkt steht.
- kann für verschiedene Projektarten genutzt werden (Modell 97 auch für Hardware-entwicklungsprojekte und mit V-Modell XT auch für Metaprojekte)
- Projektbegleitende Tätigkeiten, wie Qualitätssicherung, Projektmanagement und Konfigurationsverwaltung sind integriert.
- unterstützt auch prototypische Entwicklungen
- lässt sich anpassen und erweitern
- unterscheidet zwischen Auftraggeber- und Auftragnehmerprojekten

3.1.4 Clean Room Development

Der Begriff Cleanroom kommt aus der Fertigung von Prozessor -und Speicherchips. Dort versucht man Defekte, die häufig auch durch Verunreinigungen und Staub zustande kommen, durch Filter, Schleusentüren und Absaugungen in Reinräumen (Cleanrooms) zu reduzieren. Im Gegensatz zu Software, lassen sich defekte Teile nicht mehr reparieren. Der Aufwand für die Nacharbeitung von fehlerhafter Software kann jedoch auch sehr groß sein und es ist sehr wahrscheinlich, dass wieder neue Fehler entstehen.
[Ludewig und Lichter 2010]

IBM übernahm die Idee des Cleanrooms, also Software fehlerlos zu entwickeln und sich damit aufwändige Fehlerkorrekturen zu ersparen.

Abb. 3: Architekturmodell Cleanroom Development



[Prowell Trammell Linger Poore 1999]

Struktur

Die Struktur besteht aus 14 Aktivitäten und vier vernetzten Teilprozessen. Die Managementprozesse wirken sich auf alle Teilprozesse aus. Die Teilprozesse Spezifikation, Entwicklung und Zertifizierung werden in einer bestimmten Reihenfolge durchlaufen. Der statistische Test führt die parallel laufenden Abläufe Entwicklung und Zertifizierung wieder zusammen. Die Entwicklung erfolgt inkrementell, also in Zyklen. [Ludewig und Lichter 2010]

Cleanroom Konzepte

Abgeleitet aus Erfahrungen und Untersuchungen in den 70er Jahren wurden folgende **Regeln** abgeleitet:

- Ein großes Projekt wird in kleinere Projekte aufgeteilt, wobei jedes der Projekte mit einer überschaubaren Anzahl von Entwicklern durchgeführt werden kann.
- Der Aufwand für Analyse und Spezifikation ist höher als üblich. Es gibt keine vorgeschriebene Methode. Je nach Problem verwendet man die am besten geeigneten Verfahren und Darstellungen.
- Die Entwickler können den Code nicht kompilieren um zu testen. Man erwartet, dass die Entwickler semantisch und syntaktisch fehlerfreie Programme liefern.

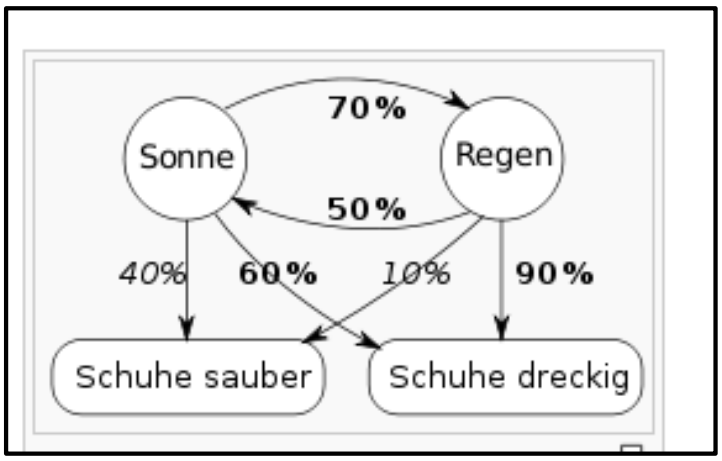
- Auf einen Test einzelner Komponenten (Unit Test) wird ganz verzichtet.
- Tests erfolgen integriert und insgesamt.
- Fehler werden nur in sehr geringem Umfang akzeptiert und von den Entwicklern selbst behoben. Fehlerhafte Komponenten werden verworfen und nicht überarbeitet.

[Ludewig und Lichter 2010]

Die Tests finden nach statistischen Methoden und basierend auf den Anwendungsfällen (Usage Models) statt. Anwendungsfälle beschreiben alle möglichen Ereignisse in einem System und ihre Wahrscheinlichkeiten des Auftretens. Häufig kommen auch Markov Modelle (benannt nach dem russischen Mathematiker A.A. Markov) und formale Grammatiken zum Einsatz. [Prowell Trammell Linger Poore 1999].

Als Beispiel für ein Markov Modell sei hier "Gefangener im Verlies" angeführt. Aussagen über das Wetter sollen auf Basis des Zustandes der Schuhe der Wärter statistisch ermittelt werden.

Abb. 4: Markov Gefangener im Verlies



[Wikipedia 2016]

CDP Cleanroom Development wurden in vielen Projekten mit kritischen, eingebetteten Systemen erfolgreich angewendet. Voraussetzung ist eine stabile Spezifikation, deshalb ist das Konzept für agile Prozesse nicht geeignet.

3.2 **Modell von Terna**

Die von Terna in der Vergangenheit praktizierte Vorgehensweise kann als klassisches Fünfphasenmodell charakterisiert werden.

Abb. 5: Projektphasen Terna

Die Projektpositionierung ist vorbereitend zu einem Projekt zu sehen und kann somit nicht als „echte“ Projektphase gesehen werden.

Analyse -und Designphase sind genau genommen zwei Phasen, die hintereinander ablaufen. Verschiedene Funktionen können aber auch parallel betrachtet werden. Das Design der Funktion wird in allen Fällen, aber erst dann festgelegt, wenn die Analyse der jeweiligen Funktion abgeschlossen ist. Bei Projekten im Pharmazie-Umfeld ist es üblich, dass in der Analysephase das vom Kunden bereitgestellte Lastenheft besprochen und in der Designphase, das sogenannte Pflichtenheft erstellt wird. In der Folge prüft der Kunde die Last gegen die Pflicht.

Als Ergebnis jeder Phase werden Dokumente erstellt. Die Fertigstellung und Abnahme der Dokumente und die erfolgreichen Tests der Software, sind Bedingung dafür, dass die nächste Phase begonnen werden kann. Der Abschluss wird als Meilenstein gesehen, eine Sitzung des Lenkungsausschusses finalisiert die Phasen formell.

Iterationsschleifen sind nicht vorgesehen! Nachträgliche Änderungen an beschriebenen Sollfunktionalitäten werden über Änderungsverfahren (Change Requests) abgewickelt.

Rahmenbedingungen für die Softwareentwicklung:

- Entwicklungen werden auf Basis von Softwarekonzepten (Designkonzepten) programmiert.
- Die Umsetzung erfolgt nach Freigabe des fertigen Konzeptes und der Kosten durch den Kunden.
- Die Erstellung von Konzepten wird nach Aufwand (Time & Material) abgerechnet.
- Entwicklungen werden dann in der Folge mit Verweis auf das Konzept zu einem Festpreis geliefert.
- Die Programmierungen erfolgen ausschließlich auf Basis von Konzepten.
- Änderungen sind nicht vorgesehen.
- Die Tests erfolgen laut den im Konzept beschriebenen Testfällen bzw. Akzeptanzkriterien.
- Der Berater bekommt das Produkt erst nach Fertigstellung zur Beurteilung.
- Der Kunde bekommt das Produkt erst nach Fertigstellung zur Beurteilung

3.3 **Bewertung**

Die bisher verwendete Methode kann als ein klassisches Vorgehensmodell betrachtet werden. Mit den fünf bzw. sechs Phasen, deren Abschluss Bedingung dafür ist, dass die nächste Phase begonnen werden kann, dem dokumentgetriebenen Ansatz und den fehlenden Iterationen kommt es dem Wasserfallmodell gleich. Die Analyse und die Designphase laufen parallel ab. Damit wurde bereits eine Adaptierung des so strikten Modelles vorgenommen. In der praktischen Anwendung gibt es immer wieder kleine Anpassungen und Abweichungen von der beschriebenen Vorgehensweise. Diese können situations- oder projektbezogen notwendig sein oder vom Kunden oder von externen, validierenden Stellen explizit gefordert werden.

Code and Fix wird, wenn auch nicht offiziell beschrieben und auch nicht gerne gesehen, bei Terna immer wieder angewandt. Leider werden auch im Nachhinein keine Dokumentationen erstellt bzw. bestehende Dokumentationen nicht aktualisiert. Bei Code and Fix wird auf eine Beschreibung der Anforderungen und auf ein Softwaredesign weitgehend verzichtet, was nicht heißt, dass nicht auch gute Software produziert werden könnte. Aufgrund des Fehlens der Dokumentation und somit der Nachvollziehbarkeit der Realisierung ist dieses Vorgehensmodell im ERP-Umfeld unbrauchbar. Der Kunde muss im Vorfeld wissen, was er bekommt und die Software muss gewartet und weiterentwickelt werden können, was ohne entsprechende Beschreibungen nicht möglich ist. Aufgrund dieser Tatsache wird die Methode nicht näher betrachtet und weiter ausgeführt.

Terna hat über die Jahre viele erfolgreiche und weniger erfolgreiche Projekte unter Anwendung der Wasserfallmethode umgesetzt. Auch viele Mitbewerber und die Hersteller, mit denen Terna Partnerverträge eingegangen ist, praktizierten und arbeiten heute noch nach diesem Modell. Die Hersteller, zu denen seitens der Implementierungspartner eine Abhängigkeit besteht, stellen entsprechende Werkzeuge zur Verfügung und empfehlen diese zu verwenden.

Die Stärken plangetriebener Methoden sind die Vergleichbarkeit und Wiederholbarkeit, die die Standardisierung mit sich bringt [Boehm und Turner 2004]. Ein wesentlicher Vorteil ist aber auch die Verständlichkeit und die eigentlich logische und leicht nachvollziehbare Abfolge der Phasen. Die Mitarbeit des Kunden ist nur am Anfang und Ende des Teilprojektes notwendig. Dies kann aus Sicht der Ressourcenbelastung als wesentlicher Vorteil der klassischen Methode gesehen werden, weil die Kunden meist Probleme in der Bereitstellung der notwendigen Kapazität haben.

In der Branche kann ein klarer Trend zu Werkverträgen beobachtet werden. Bei solchen Vereinbarungen ist es üblich, dass die Anforderungen (Lasten) und die Designs (Pflichten) so detailliert wie möglich beschrieben werden, um damit Kosten- und Terminsicherheit zu ermöglichen. In der Pharmabranche sind es die validierenden Behörden, die vorgeben, dass die Pflicht gegen die Last geprüft wird und sequentielle Abarbeitung vorschreiben. Werkverträge und die Anforderungen aus der Validierung hätten somit eigent-

lich die Anwendung klassischer Methoden als logische Konsequenz zur Folge. Dagegen sprechen die Diskussionen und Streitigkeiten, wenn durch Änderungen Mehrkosten getragen werden müssen. Im Pharmabereich spricht zusätzlich noch das aufwendige Änderungswesen bei Anpassung von Anforderungen und Designs gegen klassische Methoden.

Viele Kunden wollen Budgetsicherheit und glauben, dass sie in der Lage sind ihre Anforderungen so klar und detailliert zu beschreiben und Verträge so gestalten zu können, dass es keine Abweichungen geben kann.

In der von Terna erlebten Praxis sieht es allerdings häufig anders aus:

- Anforderungen können selten so formuliert werden, dass kein Interpretationsspielraum bleibt.
- Änderungen können sich immer wieder ergeben, vor allem bei längeren Projektlaufzeiten.
- Anforderungen sind nie vollständig beschrieben
- Änderungen ergeben sich meist erst im Detail

Neue oder geänderte, fehlende oder nicht ausreichend detaillierte Anforderungen haben immer Mehrkosten zur Folge, womit die Sicherheit eines solchen Gewerkes nur scheinbar gegeben ist.

Aus Umfragen, direkten Kundengesprächen und subjektiven Eindrücken kam die Geschäftsleitung zur Erkenntnis, dass unsere Kunden hauptsächlich wegen langer Projektlaufzeiten, Zusatzkosten für Änderungen, überschrittener Budgets und einer verbesserungswürdigen Qualität insgesamt sehr unzufrieden sind.

Der CEO[3] von Terna, Christian Kranebitter, bewertet die praktizierte Methode wie folgt:

- Die Methode ist zu träge und zu schwerfällig, sie ist zu kompliziert und für die Kunden unverständlich.
- Änderungen haben stets Zusatzaufwände und schwierige Kundengespräche zur Folge.
- Sie wird nicht einheitlich in allen Geschäftsbereichen des Unternehmens angewendet.
- In den ersten Projektphasen, Analyse und Design, ist der Nutzen für die Kunden nicht erkennbar.
- Das Team des Kunden wird zu spät in die Lösung integriert und identifiziert sich in der Folge dann damit zu wenig oder überhaupt nicht.
- Der Arbeitsaufwand steigt beim Kunden gegen Projektende, dann wenn alle Softwareanpassungen geliefert werden, massiv an. Dies führt zu Verzögerungen und Unzufriedenheit.
- Zwischen der Analyse und der Umsetzung vergeht zu viel Zeit.

[3] Chief Executive Officer. Geschäftsführer

- Es wird zu viel Dokumentation in Form von Analyse-und Designkonzepten und Listen produziert, die dann in der Folge nicht mehr verwendet und aktualisiert werden.
- Es gibt eine starke Funktionsorientierung und wenig Fokus auf die funktionierende Gesamtlösung.
- Die Budgets werden nicht zielgerichtet genutzt, was zu Überschreitungen führt.

Ob Erfolg oder Misserfolg von Projekten mit der verwendeten Methode zusammenhängt, wurde weder im Zuge dieser Arbeit noch bei Terna analysiert. Ebenso wurde nicht beurteilt, wie die Kunden das Thema klassisch oder agil sehen.

4 Agile Methoden

Im nachfolgenden Kapitel werden die Agilen Prinzipien beschrieben, die wichtigsten agilen Methoden vorgestellt und hinsichtlich einer möglichen Eignung für Terna bewertet. Dazu sollen die Beurteilung von CEO Christian Kranebitter, die Ergebnisse der Anforderungen aus den Arbeitskreisen und die besonderen Anforderungen von ERP-Implementierungspartnern (siehe 2.3) berücksichtigt werden.

4.1 Agile Methoden

Agile Methoden sind eine Folge von Rapid Prototyping und Rapid Development Erfahrungen und die Erkenntnis, dass Softwareentwicklung eher ein Handwerk ist als Ingenieurskunst. Die sich schnell ändernde internetbasierende Wirtschaft bedingt Flexibilität und Geschwindigkeit in der Softwareentwicklung. Das ist nicht unbedingt die Stärke der klassischen, plangetriebenen Methoden. Das Problem mit den Änderungen wird durch die langen Entwicklungszeiten verstärkt, sodass die Software zwar den Spezifikationen entspricht, aber häufig nicht die Erwartungen des Kunden trifft. Der Begriff „chaordic“ wurde geprägt für Arbeit, die Chaos und Ordnung auf eine Art vereinigt, die Management mittels normalen, klassischen, linearer Planung und Prozessen trotzt. [Boehm und Turner 2004]

Konkreter wird diese Sichtweise im **»Agilen Manifest«** dargestellt.

Im Jahr 2001 trafen sich führende Vertreter der Agilen Softwareentwicklungs-Bewegung in Utah (USA) und verabschiedeten das Agile Manifest. Zu den Unterzeichnern zählten neben den Scrum- Mitbegründern Mike Beedle, Ken Schwaber und Jeff Sutherland andere Größen der Szene wie Kent Beck (XP), Alistair Cockburn (Crystal), Ward Cunningham, Martin Fowler, Ron Jeffries, Andrew Hunt (Pragmatic Programmer), Robert C. Martin und Dave Thomas (Ruby on Rails).

Der Text des **Agilen Manifests** lautet:

„[Wir] haben Folgendes schätzen gelernt:
Individuen und Interaktionen - mehr als Prozesse und Tools
funktionierende Software - mehr als umfassende Dokumentation
Zusammenarbeit mit Kunden - mehr als Vertragsverhandlungen
Reaktion auf Änderungen - mehr als einen Plan zu befolgen
Obwohl auch die Dinge auf der rechten Seite ihren Wert haben, schätzen wir die auf der Linken höher ein.“ [Agile Manifesto 2001]

Die Prinzipien dahinter sind:

- Regelmäßige und frühe Auslieferung funktionierender Software
- Änderungen sind willkommen, weil sie Kundennutzen steigern
- Regelmäßig funktionierende Software liefern
- Tägliche Zusammenarbeit des Teams während des Projektes
- Vertraue dem Team und unterstütze es
- Enge Zusammenarbeit innerhalb des Teams und Selbstorganisation
- Funktionierende Software ist das wichtigste Maß für den Fortschritt
- Auftraggeber, Entwickler und Benutzer sollten ein gleichmäßiges Tempo auf unbegrenzte Zeit halten können.
- Ständiges Augenmerk auf technische Exzellenz und gutes Design
- Regelmäßige Reviews sollten Effizienz steigern
- Einfachheit - die Kunst, die Menge nicht getaner Arbeit zu maximieren - ist essenziell.

[Agile Manifesto 2001]

Bekannte Vertreter der agilen Softwareentwicklung sind:

- Scrum
- Kanban
- FDD Feature Driven Development
- XP eXtreme Programming
- Crystal
- The Eclipse Way

4.1.1 Scrum

Der wohl bekannteste Vertreter der agilen Entwicklung ist Scrum.

Scrum (engl. Gedränge) ist ein agiler Ansatz zum Management von Software-Entwicklungsprojekten und wurde von Jeff Sutherland, Ken Schwaber und Mike Beedle in den 90er Jahren entwickelt. Es basiert auf Ideen, die aus dem Kontext von Lean Production entstanden sind. Bei Scrum gibt es ein Team, das sich selbst organisiert. Scrum kann schnell erlernt werden, basiert aber darauf, dass die beschriebenen Regeln und Methoden streng eingehalten werden. [Ludewig und Lichter 2010]

In Scrum gibt es drei **Rollen** den Scrummaster, den Productowner und das Team [Ludewig und Lichter 2010].

Die Aufgabe des Scrummaster ist es, dem Team zu helfen, dass sie die Projekte richtig durchführen, sowie dafür zu sorgen, dass die Regeln eingehalten werden. Er kümmert sich darum, dass das Team störungsfrei arbeiten kann, moderiert die täglichen Meetings und hält die Dokumentation aktuell. [Ludewig und Lichter 2010]

Der Productowner vertritt die Interessen des Endkunden, kennt die Anforderungen, welche im Product Backlog gepflegt werden, plant und entscheidet welche Anforderungen in der nächsten Iteration (Sprint) umgesetzt werden, und er nimmt am Daily Scrum teil. [Ludewig und Lichter 2010]

Das Team bildet eine Gruppe von maximal zehn Entwicklern, die idealerweise am selben Standort arbeiten. Die Entwicklungstätigkeiten werden autonom durchgeführt und Entscheidung welche Aufgaben von wem durchgeführt werden, werden im Team getroffen. [Ludewig und Lichter 2010]

Die folgenden **Dokumente** werden verwendet:

Das Product Backlog enthält die Anforderungen, die im Projekt umgesetzt werden sollen. Ausgangsbasis ist das Produktkonzept oder eine Sammlung aller Anforderungen. Die Einträge sind priorisiert und der Aufwand im Team geschätzt. [Ludewig und Lichter 2010]

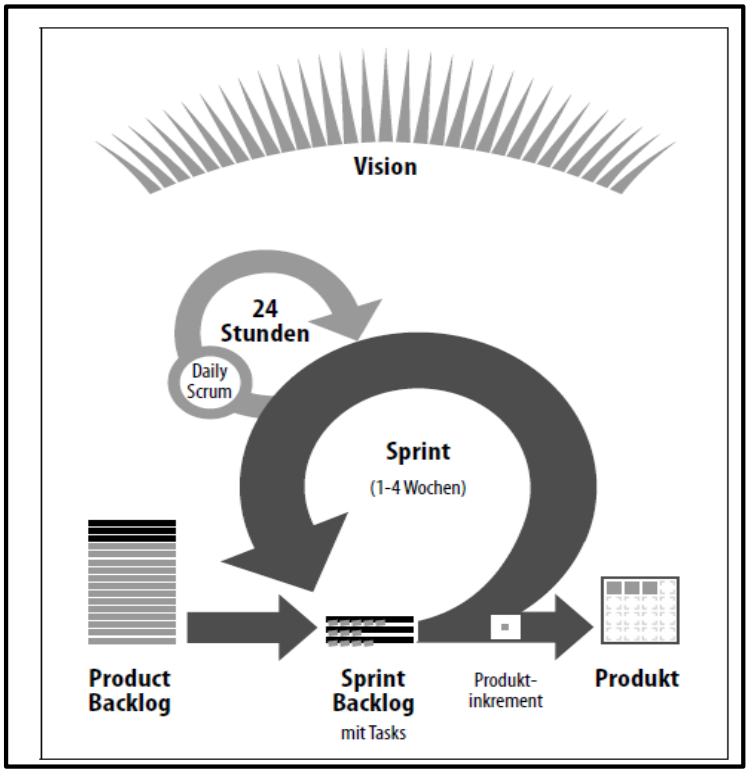
Das Sprint Backlog beschreibt die Ziele, die in einem Sprint umgesetzt werden sollen. Diese sind präzise beschrieben und in Stunden oder Tagen geschätzt. Das Sprint Backlog wird täglich aktuell gehalten und die Restaufwände eingetragen. [Ludewig und Lichter 2010]

Der Sprint Burndownbericht dokumentiert den Arbeitsfortschritt und zeigt grafisch über die Laufzeit des Sprints die Höhe der Aufwände an. Er wird täglich vom Scrummaster aktualisiert. [Ludewig und Lichter 2010]

Eine Dokumentation welche Anforderungen umgesetzt werden konnten und welche nicht und ob es Hindernisse und Probleme gegeben hat, erfolgt im Sprintendbericht. [Ludewig und Lichter 2010]

Der Prozess:

Abb. 6: Scrum Prozess



[Dräther Koschek Sahling 2013]

Scrum verlangt einen iterativen Prozess, das Produkt entsteht dabei inkrementell.

Am Anfang werden die Anforderungen in einem Anforderungsworkshop priorisiert und bewertet und im Product Backlog festgehalten. Die Anforderungen, die in einem Sprint umgesetzt werden, werden in das Sprint Backlog übernommen. Ein Sprint hat eine fixe Zeit, z.B. zwei Wochen. Es gibt ein tägliches, kurzes Abstimmungsmeeting, das sogenannte Daily Scrum, das max. 15 Minuten dauert und in dem das Sprint Backlog und der Burn Down Bericht aktualisiert werden. Am Ende jeden Sprints gibt es zwei Bewertungsrunden, den Sprint Review und die Sprint Retrospektive. Im Sprint Review wird festgestellt, ob alle Anforderungen vollständig und in der geforderten Qualität umgesetzt wurden und in einem Sprintenbericht dokumentiert. Im Sprint Retro wird beurteilt, wie gut der Scrumprozess umgesetzt wurde und mögliche Verbesserungsmaßnahmen geplant. [Ludewig und Lichter 2010]

Scrum ist heute sehr verbreitet. Laut einer Studie der Hochschule Koblenz aus dem Jahre 2014 ist Scrum mit 86% die meistbenutzte agile Methode. Danach folgen Kanban, XP und Feature Driven Development. [Komus 2014]

4.1.2 Kanban

Hinter der Idee von Kanban steht, dass die Anzahl der in Arbeit befindlichen Arbeitspakete (WIP Work in Progress) limitiert werden soll. Man spricht von WIP-Limits. Es sollen nur dann neue Arbeitspakete ein gelastet werden, wenn bestehende Arbeitspakete erledigt wurden. Die Kanban Karte (jap. Kanban = Signalkarte) dient dazu als Signal. [Kniberg 2010]

Prinzipien
Kanban kommt ursprünglich aus der Automobilindustrie, im Speziellen vom Toyota Production System (TPS). Der schonende Umgang mit knappen Ressourcen hat die beiden Vordenker Taiicho Ohno und Kiichiro Toyota dazu bewogen ganz neue Überlegungen anzustellen. Sie haben sich auf den Fluss des Produktes durch den Produktionsprozess konzentriert und mitunter festgelegt, dass nur das produziert wird, was wirklich gebraucht wird und das zu dem Zeitpunkt zu dem es gebraucht wird. Verschwendung soll vermieden werden. [Leopold und Kaltenecker 2012]

Laut Kaltenecker [Leopold und Kaltenecker 2012] wurden drei Arten der Verschwendung definiert:

- Muda Aufgaben, die Ressourcen verschwenden aber keinen Wert liefern
- Mura Unregelmäßigkeiten im Produktionsprozess
- Muri Überlastung

Kernelement der Produktionsablaufsteuerung sind die Kanban (Signalkarten). Mit diesen Signalkarten wird angezeigt, dass eine Tätigkeit fertiggestellt ist und Nachschub benötigt wird. Man spricht hier auch vom Holprinzip (Pull-System). Dadurch werden Bestände (Bestände sind Verschwendung) reduziert und Probleme werden durch erkennbare Staus in den Vorwerkstätten sofort sichtbar. Die zur Verfügung stehenden Signalkarten sind limitiert, damit kann nicht mehr Arbeit in das System ein gelastet werden. David J. Anderson ist bei seinen Überlegungen ursprünglich auch vom Drum-Buffer-Rope- Konzept in der Theory of Constraints von Eliyahu M. Goldratt ausgegangen, das beschreibt, dass der Engpass den Durchsatz bestimmt, dass jedes System spezifische Engpässe hat und dass die Engpässe sich ändern können [Goldratt 1999]. David J. Anderson hat Kanban für die Softwareentwicklung adaptiert [Leopold und Kaltenecker 2012].

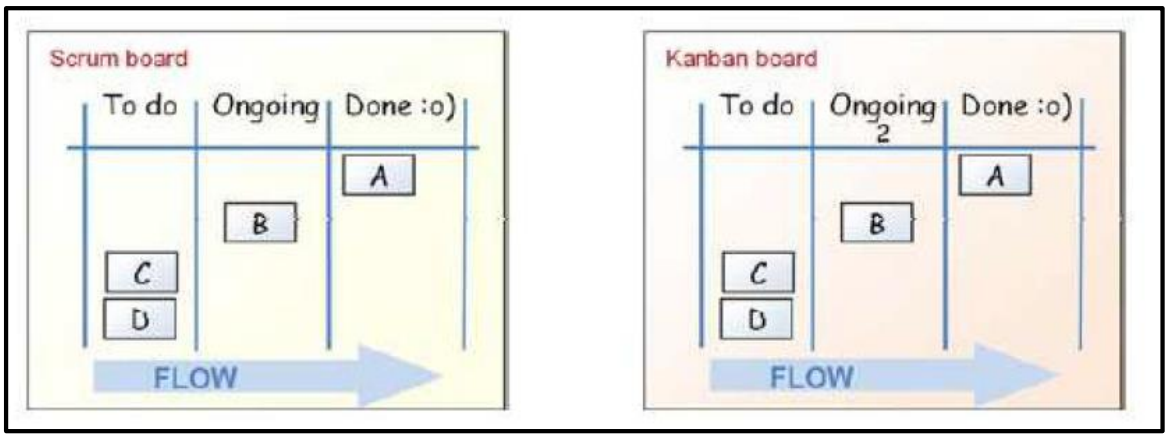
Wesentliche Unterschiede zu Scrum sind laut Kniberg [Kniberg 2010]:

- Scrum ist standardisierter und gibt mehr Grenzen vor
- Kanban lässt fast alles offen, außer; Visualisieren Sie den Arbeitsablauf und Beschränke die Arbeitspakete (Limit WIP…Work in Progress)

- Eine Mischung der Werkzeuge macht oft Sinn, bei Scrum sind aber grundlegende Werkzeuge zu verwenden
- Bei Scrum gibt es drei Standardrollen, bei Kanban ist keine feste Rolle vorgeschrieben.
- Scrum schreibt Iterationen in einem festen Zeitrahmen vor.
- Scrum begrenzt WIP pro Iteration, Kanban pro Zustand; Ongoing 2 bedeutet es dürfen nur 2 Arbeitspakete „Ongoing“ sein.

[Kniberg 2010]

Abb. 7: Scrum vs. Kanban



[Kniberg 2010]

- Scrum lässt innerhalb einer Iteration keine Änderungen zu (neue Arbeitspakete)
- Ein Scrumboard wird zwischen zwei Iterationen geleert, in Kanban kann es bestehen bleiben.
- Ein Scrumboard gehört genau einem Team, in Kanban kann es auch funktionsübergreifende Teams geben.
- Scrum Arbeitspakete müssen in einen Sprint passen. Diese Vorschrift gibt es bei Kanban nicht.
- Scrum schreibt Schätzen und Geschwindigkeit vor.

[Kniberg 2010]

4.1.3 FDD Feature Driven Development

FDD wurde von Jeff de Luca 1997 definiert und somit schon vor dem Agilen Manifest [Agile Manifesto 2001].

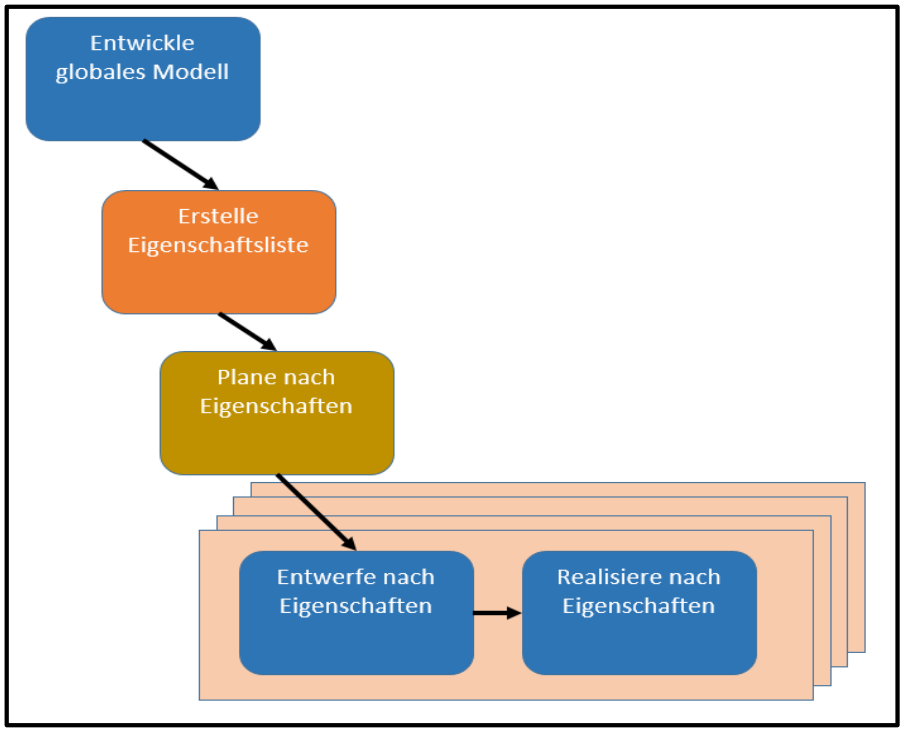
FDD stellt die Eigenschaften eines Systems (eng. Features) in den Vordergrund. Die Wurzeln liegen in der Organisation eines Großprojektes mit 50 Entwicklern, weshalb FDD für Großprojekte geeignet ist. Aufbauend auf diesen Erfahrungen wurde FDD als iterativer Ansatz entwickelt. [Bunse und Knethen 2008]

Es werden fünf Prozesse durchlaufen:

- FDD Prozess 1: Entwickle ein globales Modell
- FDD Prozess 2: Erstelle eine Eigenschaftsliste (Feature liste)
- FDD Prozess 3: Plane je Eigenschaft
- FDD Prozess 4: Entwerfe nach Eigenschaften
- FDD Prozess 5: Realisiere nach Eigenschaften

[De Luca 1998]

Abb. 8: FDD Vorgehensmodell



[Bunse und Knethen 2008]

Die ersten drei Aktivitäten (Kernaktivitäten) sind sequentiell organisiert, Entwurf und Realisierung iterativ organisiert und agil definiert. Die Aktivitäten basieren auf den Eigenschaften (Features). Es werden kleine, für den Kunden sichtbare, Funktionen dargestellt. Eigenschaften müssen dabei innerhalb von maximal zwei Wochen realisiert werden. Es

erfolgt auch vorher eine entsprechende Priorisierung und auf dieser Basis eine Planung für die folgenden Iterationen. [Bunse und Knethen 2008]

In FDD ist der Kunde als Fachexperte in einer Schlüsselrolle und prüft dann das Ergebnis der realisierten Eigenschaften (Features). [Bunse und Knethen 2008].

FDD lässt sich leicht in bestehende Organisationsstrukturen integrieren und ist auch für die Entwicklung kritischer Systeme geeignet [Bunse und Knethen 2008]. Die Methode ist jedermann zugänglich und auf zehn Seiten beschrieben.

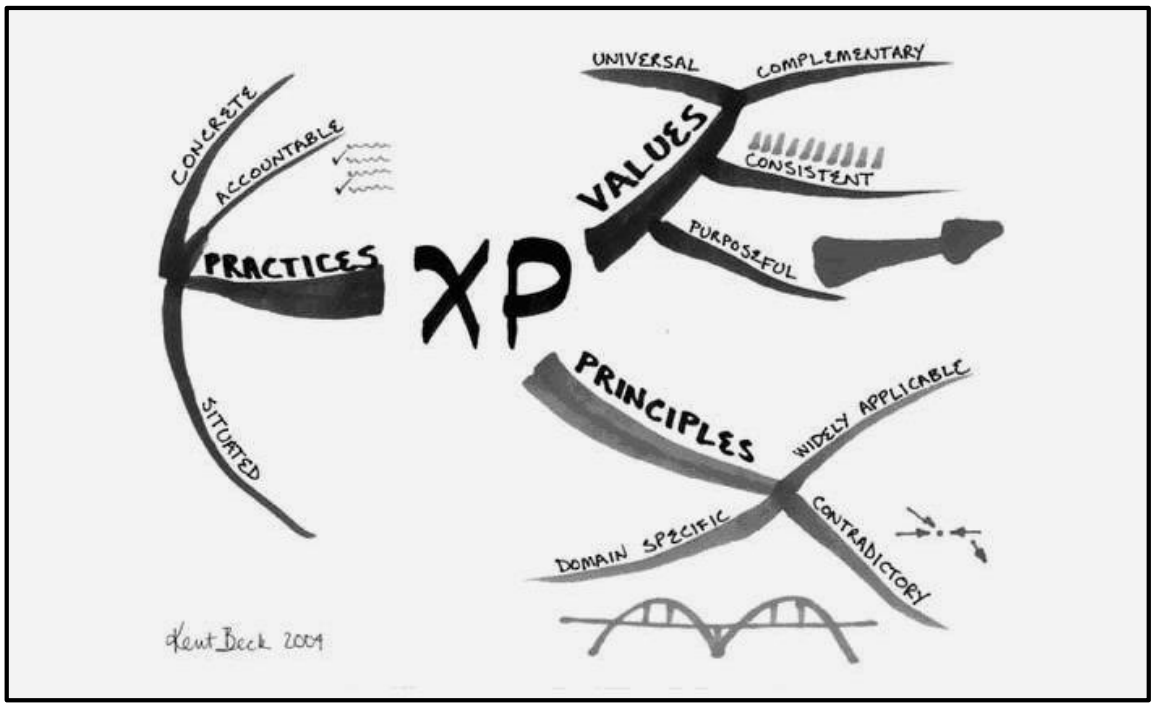
FDD wurde nachweislich schon mit Erfolg bei großen und kritischen Projekten eingesetzt [De Luca 1998].

4.1.4 XP eXtreme Programming

Die Ideen zu eXtreme Programming stammen von Kent Beck, einem Branchenguru für objektorientiere Softwareentwicklung. XP besteht aus einer Reihe einfacher, aber eng zusammengehörender Praktiken. XP ist ein sehr strenger und disziplinierter Entwicklungsprozess. XP eignet sich für kleinere und mittlere Teams bis zu 15 Personen. Iterationen in XP dauern etwa zwei Wochen. XP stellt die Menschen und nicht Dokumente, Werkzeuge, Prozesse in den Mittelpunkt.
[Hruschka, Peter; Rupp, Chris; Starke, Gernot 2009]

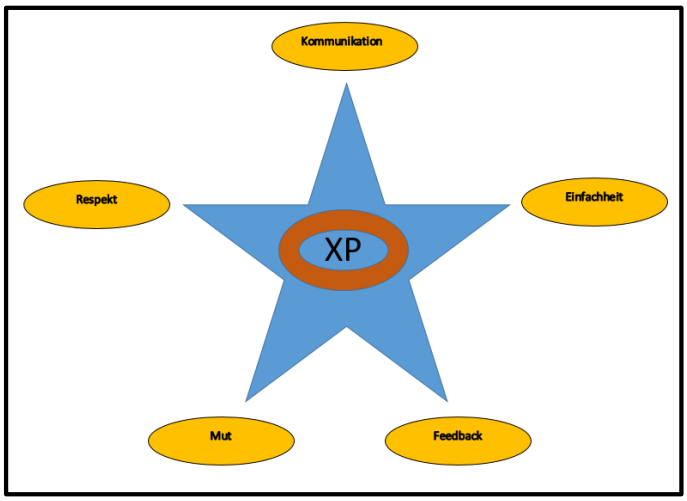
XP ist ein Rahmenwerk bestehend aus den Bestandteilen Werte (Values), Prinzipien (Principles) und Techniken (Practices). [Beck 2004]

Abb. 9: values, principles, practicals



[Beck 2004]

Abb. 10: Werte von XP



Prinzipien von XP

- Unmittelbares Feedback – gute Lerneffekte
- Einfachheit anstreben – bessere Verständlichkeit, schnelles Feedback
- Inkrementelle Veränderung – kontinuierlich kleine Veränderungen
- Qualitätsarbeit – Erhöhung der Zufriedenheit
- Lernen lehren - Selbstentscheidung
- Geringe Anfangsinvestition – Konzentration auf das Wichtigste
- Auf Sieg spielen – als Einstellung in XP im Gegensatz zu fehlerfrei
- Offene, aufrechte Kommunikation
- Instinkte Nutzen – Vertrauen
- Verantwortung übernehmen – nicht zuweisen
- An örtliche Gegebenheiten anpassen – Punkte aus dem Lehrbuch lassen sich nicht immer durchsetzen
- Mit leichtem Gepäck reisen – nicht zu viele Werkzeuge und Methoden vorgeben
- Ehrliches Messen

[Neus, Trompeter, Mandischer 2011]

Konzepte „ohne Doppelpunkt

- Managementkonzepte-integrales Team, Planungsspiel, kurze Release Zyklen, Standup Meeting, Rückblick
- Teamkonzepte-gemeinsame Codeverantwortung, Codier Richtlinien, erträgliche Arbeitsbelastung, zentrale Metapher, laufende Integration
- Programmierkonzepte-testgetriebene Entwicklung, Strukturverbesserung (Refac-turing), einfacher Entwurf, Paar-Programmierung

[Ludewig und Lichter 2010]

Rollen in XP

In Wesentlichen beschränkt sich XP auf drei Rollen: Kunde, Entwickler und XP-Coach. Von essentieller Bedeutung ist die permanente Präsenz des Kunden. Diese Rolle kann auch ein Vertreter des Kunden in räumlicher Nähe, ein sogenannter onsite customer übernehmen. Die Kunden nehmen an der Iterationsplanung teil und klären alle geschäftlichen Fragen. Der XP-Coach unterstützt das Entwicklungsteam und stellt Fragen um Entscheidungen herbei zu führen. [Hruschka, Peter; Rupp, Chris; Starke, Gernot 2009]

Planen mit XP

In der Softwarebranche funktionieren Planungen über längere Zeiträume in der Regel nicht, was auch ein Grund für viele gescheiterte Projekte ist. Bei XP wird jede Iteration auf Basis der Erfahrungen der vorigen Iteration geplant. Pläne, die weit in die Zukunft gehen, werden vermieden. XP beteiligt das gesamte Entwicklungsteam an der Iterationsplanung. [Hruschka, Peter; Rupp, Chris; Starke, Gernot 2009]

Automatisiertes Testen

Durch Unit Testing wird für dauerhaft stabilen Programmcode gesorgt, was auch das Vertrauen des Teams in die eigene Arbeit stärkt. [Hruschka, Peter; Rupp, Chris; Starke, Gernot 2009]

Häufige Integration

Nicht nur der Programmcode wird getestet, es wird auch regelmäßig (z.B. täglich und automatisiert) eine lauffähige Testversion erstellt. [Hruschka, Peter; Rupp, Chris; Starke, Gernot 2009]

4.2 Anforderungen an neue Methode

Auf Basis der Bewertung von CEO Christian Kranebitter wurden in Arbeitskreisen die nachfolgenden Problemkreise identifiziert, weitere Schwerpunkte ergänzt und daraus Anforderungen an eine neue Methode erstellt, die deutliche Verbesserungen bringen sollte.

4.2.1 Kunde

Kundenintegration

Die Kunden kommen mit der aktuellen Methode zu spät mit der Software in Berührung, sind nicht geschult und nicht zuletzt aus Gründen des Unverständnisses mit den Produkten unzufrieden. Dies wirkt sich negativ auf die Identifikation mit der Anwendung aus.

Nutzen erkennen

Für den Kunden ist der Nutzen der Analyse -und Designphase unklar. Dies kommt mitunter auch daher, dass die Themen zu allgemein sind und die Kunden häufig dieselben Fragen mehrfach beantworten müssen. In der Folge entsteht dann auch noch großer Aufwand mit Dokumentation und Prüfung dessen was dokumentiert wurde. Während dieser Zeit kann die Software nicht genutzt werden. Die Themen werden in scheinbar wahlloser Reihenfolge, ungeachtet deren Wichtigkeit und Beitrag zur Wertschöpfung, behandelt.

Anforderungen

Die neue Methode soll den Kunden besser und früher integrieren. Durch frühes und ständiges Arbeiten mit der Software soll besseres Verständnis und dadurch mehr Identifikation mit den Produkten geschaffen werden. Die Analyse -und Designphase soll es auf dieser Ebene nicht mehr geben. Die einzelnen Anforderungen sollen im Detail analysiert und aus Gründen der Verständlichkeit in der Sprache des Kunden beschrieben werden. Entscheidend dabei ist, dass für den Kunden der Nutzen klar erkennbar ist.

4.2.2 Anforderungen (Requirements)

Fehlendes Konzeptverständnis

Die Kunden haben häufig nicht die Fachleute und die Erfahrung, dass sie ein technisches Feinkonzept verstehen und in der Folge auch freigeben können. Dies kann aber auch daran liegen, dass das Konzept nicht so geschrieben ist, dass man es verstehen kann und dass der Berater, der das Konzept erstellt hat, den Kunden nicht verstanden hat. In der Folge versteht dann der Anwendungsentwickler den Berater nicht. Ist das Konzept abgenommen gibt es in der Regel keine Prüfungen, ob die Zwischenprodukte auch dem entsprechen, was der Kunde eigentlich wollte.

Dokumentation

Es wird zu viel Zeit und Aufwand in die Erstellung von Dokumenten investiert. Im Speziellen gemeint sind hier wieder die Analyse-und Designdokumente, die in der Folge dann nicht mehr verwendet werden, was wiederum mit dem Zeitpunkt der Erstellung und der Aktualität zusammenhängt.

Beurteilung Teilergebnisse/Zwischenprodukte

Es gibt bis zur Bereitstellung zum Test der fertigen Lösung keine Möglichkeit Missverständnisse rechtzeitig zu erkennen und zu beseitigen, weder zwischen Programmierer und Berater, noch zwischen Berater und Kunde.

Einzelfunktion versus Gesamtfunktion

Die Tests beziehen sich zwischen den Meilensteinen nur auf Einzelfunktionen. Integrationstests finden nur bei Meilensteinen statt. Auch in der Realisierung der Funktionalisierung werden nicht Gesamtprozesse betrachtet, sondern die Prozesse einzelner Bereiche, sodass eine funktionierende Gesamtlösung häufig nicht präsentiert werden kann.

Anforderungen

Die Anforderungen sollen in einer für den Kunden verständlichen Sprache beschrieben werden. Die Formulierung von Akzeptanzkriterien soll zu besserem Verständnis über die gewünschten und geforderten Funktionalitäten und Ergebnissen bei Kunde, Entwickler und Berater führen. Die Analyse -und Designdokumente in der jetzigen Form sollen abgeschafft und durch eine Lösungsbeschreibung ersetzt werden. Die Lösungsbeschreibung wird nach Lieferung des Produktes erstellt. Die Zwischenprodukte sollen dem Berater und dem Kunden zur Beurteilung vorgelegt werden, um Missverständnisse und Änderungen bereits früher feststellen zu können. Durch kleinere Teams, die eng zusammenarbeiten und integriertes Betrachten und Testen von Gesamtprozessen, sollen funktionierende Gesamtlösungen gezeigt werden können.

4.2.3 Änderungen

Änderung der Anforderungen

Nachdem zwischen Beauftragung und Test keine Interaktion mit dem Kunden und Berater erfolgt, gibt es keinen Abgleich mit der Aktualität der Anforderungen. Dieses Problem gibt es speziell bei Projekten mit längeren Durchlaufzeiten.

Mehrkosten

Müssen gelieferte Programme geändert werden, so ist dies mit einem verhältnismäßig großen Aufwand verbunden. Die Mehrkosten oder zumindest Teile davon, tragen entweder der Lieferant oder der Kunde, was immer auf beiden Seiten zu großen Diskussionen und zu Unzufriedenheit führt.

Anforderung

Durch bessere und frühere Integration des Kunden, klare Beschreibung der Anforderungen, regelmäßige Lieferung von getesteten, dokumentierten Inkrementen sollen Änderungen früher erkannt werden. Damit sollen die Änderungskosten deutlich reduziert werden.

4.2.4 Projektmanagement

Fortschrittsbeurteilung/Teilbarkeit

Der Programmierauftrag wird als Ganzes gesehen, weshalb keine objektive Beurteilung des Arbeitsfortschritts für den Entwickler und dem Projektmanagement möglich ist. Es ist auch kaum möglich dem Entwickler andere Aufträge zu geben, weil es keine abschließbaren Teilaufträge gibt. Nachdem die Aufträge nicht geteilt werden, ist es auch sehr schwierig die Auslastung in der Entwicklungsabteilung auszugleichen. Dies kann dazu führen, dass die Arbeiten von einzelne Personen in Überstunden erledigt werden müssen, was immer zu Stresssituationen und unzufriedenen Mitarbeitern führt.

Zielgerichtete Nutzung des Budgets

Bei Gegenüberstellungen der verbrauchten Budgets im Vergleich zum gelieferten Kundennutzen wurde festgestellt, dass zu Beginn zu viel Zeit für Tätigkeiten verbraucht wird, die zu wenig Mehrwert für den Kunden bzw. das Projekt liefern. Gegen Projektende ist das Budget häufig aufgebraucht, was wieder zu Frust beim gesamten Projektteam führt.

Trägheit und Schwerfälligkeit

Die starre Struktur und die Standardisierung der Abläufe haben entsprechend lange Durchlaufzeiten zur Folge. Entscheidungen werden zu spät oder gar nicht getroffen.

Verständlichkeit der Methode

Die aktuelle Methode scheint oberflächlich betrachtet logisch zu sein, im Detail gelingt es aber häufig nicht die Vorgangsweise dem Kunden, aber auch den eigenen Mitarbeitern verständlich zu machen. Dies liegt mitunter auch am Fehlen eines Handbuchs und einer nicht einheitlichen Anwendung der Methode.

Risiko

Das betriebswirtschaftliche Risiko soll durch mehr Übersicht, klare Akzeptanzkriterien und durch nachvollziehbare Tests von Inkrementen, Einzelfunktionen und Gesamtprozessen reduziert werden.

Anforderungen

Durch das „Schneiden“ von Arbeitspaketen in kleinere Einheiten soll ein besserer Überblick über die Arbeitspakete geschaffen werden. Durch kleinere Arbeitspakete soll in der Entwicklung kürzer und feiner geplant werden und sowohl Unterbrechungen, als auch Mehrarbeit deutlich reduziert werden.

Durch ständige Priorisierung der Arbeitspakete unter Berücksichtigung des Kundennutzens soll sichergestellt werden, dass die wichtigsten Funktionalitäten zuerst umgesetzt werden.

Durch kleinere Projektteams, die eng zusammenarbeiten und sich selbst organisieren, sollen Entscheidungen wesentlich schneller getroffen werden können.

Das betriebswirtschaftliche Risiko soll durch mehr Übersicht, klare Akzeptanzkriterien und durch nachvollziehbare Tests von Inkrementen, Einzelfunktionen und Gesamtprozessen reduziert werden

4.2.5 Allgemein

Zusammenarbeit/Wissenstransfer

Es erfolgt kein regelmäßiger Wissenstransfer zwischen den Entwicklern, sowie zwischen Entwicklern und Beratern.

Hardware

Die ERP-Hersteller mit denen Terna Partnerverträge eingegangen ist, bieten die Software auch als Service in der Cloud an. Terna bietet an die Software in einem Rechenzentrum zu betreiben (Private Cloud).

Umfeld Software

Hauptsächlich werden Anpassungen und Erweiterungen zur vom Hersteller gelieferten Anwendung entwickelt. Es gibt aber auch Standardentwicklung in kleinerem Umfang.

Anforderungen

Durch kleine Projektteams, die sich in erster Linie um das eine Projekt kümmern, soll die Zusammenarbeit und das Gesamtverständnis gefördert und entwickelt werden. Die Methode soll auch für die Entwicklung von Standardsoftware in der Private Cloud geeignet sein.

4.3 Bewertung agiler Methoden

In diesem Kapitel wird bewertet, ob die Anforderungen aus Kapitel 4.2 mit den vorgestellten Methoden erfüllt werden können. Wenn hier von Methoden oder agilen Methoden gesprochen wird, so gilt das für die vier vorgestellten Methoden und nicht unbedingt für agile Methoden im Allgemeinen.

4.3.1 Kunde

Die Einbindung und Integration des Kunden ist bei XP fest vorgeschrieben, er ist ständiges Mitglied. Bei Scrum ist die Rolle des Kunden nicht vorgesehen, er wird durch den Productowner vertreten. Eine regelmäßige Abstimmung mit dem Kunden soll dessen Einbindung sicherstellen. Bei Kanban gibt es keine vorgeschriebenen Rollen, es wäre aber jederzeit möglich ihn zu integrieren mit der Einschränkung, dass neue Rollen nur dann eingeführt werden, wenn diese auch Mehrwert bringen können [Kniberg 2010]. Bei FDD ist der Kunde einer der Fachexperten beim Entwickeln des globalen Modells und prüft dann gegen die realisierte Eigenschaft, dazwischen gibt es keine Abstimmungen.

Analyse und Design gibt es im agilen Umfeld genauso, aber nicht strikt hintereinander und als abgeschlossene Phasen, sondern auf funktionaler Ebene und iterativ. In FDD finden

die Analysen- und Designphasen jedoch sequentiell und somit ähnlich dem Wasserfallmodell statt.

Bei agilen Methoden, von FDD abgesehen, sind keine abgegrenzten Designphasen vorgesehen. Eine Einbindung des Kunden, damit eine frühe Identifikation mit der Software möglich ist, ist jedoch nur bei XP fix vorgeschrieben, bei Kanban optional, bei Scrum nur indirekt, durch Abstimmung außerhalb des Teams möglich. Eine der wesentlichen Bedingungen für eine erfolgreiche Integration des Kunden ist, dass die Anforderungen in seiner Sprache diskutiert und dokumentiert werden. Dies gilt im Übrigen auch für die Akzeptanzkriterien.

Daraus lässt sich ableiten, dass agile Methoden mit iterativen Analyse- und Designphasen und jene Methoden, die den Kunden aktiv integrieren, die von Terna erstellten Anforderungen von Terna erfüllen. Voraussetzung dafür ist, dass die Anforderungen und in Kundensprache beschrieben werden. Es wurde nicht bewertet, ob eine regelmäßige Abstimmung mit dem Kunden, wie es mit Scrum üblich ist, dieselben Ergebnisse bringen könnte.

4.3.2 Anforderungen (Requirements)

Nachdem die Beschreibung der Anforderung und Akzeptanzkriterien in allen angeführten Methoden in der Sprache des Kunden erfolgt, kann man davon ausgehen, dass Kunde und Berater besser verstehen können was gewünscht wird. Es ist aber auch in klassischen Methoden nicht ausgeschlossen die Anforderung in dieser Form zu beschreiben. Eine technische Beschreibung der Anforderung ist für die Aufwandsbetrachtung und für die Realisierung in der Programmierung ein Vorteil. Es gibt aber kaum Kunden, die in der Lage sind, den technischen Teil zu verstehen und auf Richtigkeit zu prüfen, deshalb kann man auch darauf verzichten.

Es kann deshalb angenommen werden, dass mit agilen Methoden kein besseres Konzeptverständnis erwirkt werden kann. Eine klare und einfache Beschreibung von Anforderung und Akzeptanzkriterien sind Bedingung dafür, dass Konzepte von allen Beteiligten verstanden werden können. Dies ist völlig unabhängig davon, ob klassisch oder agil entwickelt wird.

Bei allen vorgestellten Methoden ist ein Verständnis der Anforderungen notwendig, damit die richtigen Funktionalitäten geliefert werden können. Bei agilen Methoden erfolgt die Beschreibung der Anforderung in der Regel in Form von Userstories [Rupp 2014] und ist somit ein wichtiger Teil der Dokumentation. In der einschlägigen Literatur wird nicht darauf hingewiesen, dass Dokumentation in agilen Projekten einen geringeren Stellenwert hat, als in klassischen. Bei XP wird allerdings aufgrund der Kundennähe weitgehend auf Dokumentation verzichtet.

Die Anforderung von Terna unnötige, umfangreiche, budgetvernichtende Dokumentation zu vermeiden, bezieht sich darauf, dass nicht von Beginn an alle Details der Anforderung

und der Realisierung (Design) beschrieben werden sollen. Schritt für Schritt müssen die fertiggestellten, abgenommenen Eigenschaften dann entsprechend dokumentiert werden. Gerade aber bei Fixpreisprojekten wird man auf eine Dokumentation vor Umsetzung nicht verzichten können, weil diese auch Basis für den Vertrag ist.

Neben der Dokumentation, dient eine Beurteilung von Zwischenergebnissen anhand von funktionierender Software mitunter dazu, dass geprüft werden kann, ob die Anforderung von Berater und Programmierer verstanden wurde. Damit soll auch sichergestellt werden, dass am Ende des Projektes das Produkt funktioniert und aussieht, wie es der Kunde benötigt. Nachdem die Software als solches funktionsfähig und dokumentiert ist, ist eine Beurteilung durch Berater und Kunde einfach möglich. Fehlerfreiheit durch getestete Software wird einen wichtigen Beitrag zur Akzeptanz bringen. Das kaufmännische Risiko, dass am Ende des Sprints wieder alles verworfen werden muss besteht, ist aber durch das frühe Erkennen wesentlich geringer als am Ende des Projektes. Ob der Kunde dann immer zur Verfügung steht um die Zwischenergebnisse zu testen, wird die Praxis zeigen. Auch bei beim Projektteam sind entsprechende Voraussetzungen hinsichtlich Verfügbarkeit zu schaffen.

Ob Einzel - oder Gesamtfunktionalitäten bei der Lieferung der Inkremente getestet werden, wird im Detail in den Methoden nicht beschrieben. Es ist Aufgabe des Projektmanagements sicherzustellen, dass Komplettprozesse ausreichend getestet werden.

4.3.3 Änderungen

Bei agilen Vorgehensmodellen sind Änderungen und Feedback aktiv erlaubt und werden nach Möglichkeit bei der nächsten Iteration berücksichtigt, deshalb ist auch kein Change Request Verfahren beschrieben. Eine Ausnahme bei den betrachteten Methoden stellt FDD dar, da hier die Kernaktivitäten sequentiell ablaufen und nur Entwurf und Realisierung iterativ sind [Bunse und Knethen 2008]. In klassischen Methoden werden Änderungen als Ausnahme behandelt.

Speziell bei längeren Projekten können sich die Anforderungen ändern, z.B., weil sich das Geschäftsumfeld des Kunden geändert hat, oder aufgrund von Management Entscheidungen, die auch Einfluss auf Funktionalitäten haben. Es ist sicher von Vorteil, wenn von Anfang an organisatorisch und im Prozess berücksichtigt wird, dass Änderungen möglich sind. Damit können die Kosten in einem vernünftigen Maß gehalten werden. Werden aber Kosten und Termine wesentlich beeinflusst, so können diese, unabhängig davon, ob die Projekte nach Aufwand oder zu einem Fixpreis verkauft wurden, nicht so einfach akzeptiert werden.

Durch regelmäßige Interaktion mit dem Kunden soll sichergestellt werden, dass keine unnötigen Softwareteile programmiert werden. Durch frühes Erkennen von Abweichungen können die Kosten wesentlich reduziert werden. Änderungen, die einen wesentlichen Mehraufwand zur Folge haben, müssen aber natürlich auch unter kaufmännischen Aspek-

ten behandelt werden. Fixpreise sind hier eher ein Problem. Hier wurden Vertragsmodelle, wie der agile Festpreis [It-Agile 2013] entwickelt, in denen das Vorgehen und der Umgang mit Mehr –und Minderkosten geregelt werden.

Es lässt sich ableiten, dass geänderte Anforderungen und Festpreise, aber auch Einhaltung eines Projektbudgets, eigentlich ein Widerspruch sind und agile Vorgehensmodelle dafür keine wesentlichen Verbesserungen bringen. Kleine Änderungen können ohnehin auch bei klassischen Methoden realisiert werden. Die iterative Vorgehensweise ist aber geeignet, die Kosten gering zu halten. Wenn Budget oder Preis trotzdem eingehalten werden müssen, dann geht das zwangsläufig zu Lasten von Funktionalität an anderen Stellen.

4.3.4 Projektmanagement

Das Bilden und Verwalten von kleinen, überschaubaren Inkrementen und die regelmäßige Fortschrittsbeurteilung liefern einen wesentlichen Beitrag zu mehr Übersicht. Inkremente sind aufgrund ihrer Größe in kurzer Zeit abschließbar. Deshalb können schneller andere Funktionalitäten eingesteuert werden, ohne den Entwickler durch Unterbrechung zu stören. Auch Änderungen werden, sofern nicht der aktuell in Arbeit befindliche Teil betroffen ist, nicht als solche wahrgenommen. Alle vorgestellten Methoden sind dazu geeignet Arbeitspakete oder Funktionalitäten zu „Schneiden“ (in kleinere Einheiten zu unterteilen). Die technischen Möglichkeiten und organisatorischen Voraussetzungen müssen dabei natürlich auch berücksichtigt werden. Laufende Priorisierung soll sicherstellen, dass die wichtigeren Funktionalitäten vor den unwichtigeren umgesetzt werden. Das ist bei allen betrachteten Methoden möglich. Gemeinsam ist Ihnen auch die Forderung nach kleinen, sich selbst organisierenden Teams, die schnell Entscheidungen treffen können.

Das kaufmännische Risiko kann durch fehlerfreie Programmierung und entsprechende Tests reduziert werden. Eine entsprechende Spezifikation wird vorausgesetzt. Es ist sicherzustellen, dass nicht nur Einzelfunktionen, sondern auch die Gesamtfunktionalitäten getestet werden, was aber eher Aufgabe des Projektmanagements oder der vergleichbaren in den Methoden beschriebenen Rollen ist.

Bei der Einführung und Anwendung von agilen Methoden wird auf die Wichtigkeit von Tests und das Vorhandensein entsprechender Test-und Akzeptanzkriterien an vielen Stellen hingewiesen. Tests müssen unabhängig von der Methode immer an die Anforderungen und das Risiko angepasst werden. Es lässt sich nicht ableiten, dass agile Methoden nicht für ERP-Implementierungspartner aufgrund des betriebswirtschaftlichen Risikos geeignet sind. Bei konsequenter Anwendung können agile Methoden sogar das Risiko reduzieren.

4.3.5 Allgemein

Durch kleine Teams, die eng zusammenarbeiten und die sich zum Großteil ihrer Arbeitszeit nur um das eine Projekt kümmern, kann ein optimaler Austausch von Wissen und Erfahrung stattfinden.

Ob die Hardware beim Kunden vor Ort oder in einem Rechenzentrum betrieben wird, hat nur technische Auswirkungen, aber keinen Einfluss auf das Vorgehensmodell und wird somit nicht näher betrachtet.

Partner liefern hauptsächlich kundenspezifische Erweiterungen für Standardsoftware, liefern aber auch eigene Anwendungen und Branchenlösungen, somit kann man von Standardentwicklung sprechen. Es lassen sich in der Literatur keine Hinweise dafür finden, dass agile Methoden nur für Standardentwicklung geeignet wären und es lassen sich auch keine logischen Gründe dafür ableiten, weshalb angenommen werden kann, dass das Softwareumfeld keinen Einfluss auf die Eignung hat.

4.4 Zusammenfassung

4.4.1 Eignung agiler Methoden

Die Bewertung der im Kapitel 4.2 beschriebenen Terna-spezifischen Anforderungen wurden Kriterien, wie Wichtigkeit und Relevanz, sowie Erfüllungsgrad der Anforderungen berücksichtigt. Es wurde nicht bewertet, ob die Methoden auch zu Verschlechterungen in anderen Bereichen führen könnten. Ebenso nicht berücksichtigt wurden die zu schaffenden Voraussetzungen seitens Terna und deren Kunden und die Probleme, die bei der Umstellung auftreten können.

Neben Scrum wurden die Methoden Kanban, XP eXtreme Programming und FDD Feature Driven Development als bekannte Vertreter von Agilen Methoden vorgestellt und hinsichtlich ihrer Eignung für Terna bewertet. Bei dieser Bewertung sind auch Kriterien, wie Wichtigkeit und Relevanz berücksichtigt worden.

Scrum ist der am meisten verbreitete Vertreter der agilen Methoden [Komus 2014]. Scrum schreibt eine konsequente Anwendung des Modells vor. Eine direkte Integration des Kunden ist nicht vorgesehen. Auf alle Fälle spricht die Bekanntheit und praktische Erfahrung bei Kunden und Mitarbeitern dafür Scrum zu verwenden. Die fehlende direkte Kundenintegration, sowie die Starrheit in der Anwendung sind Argumente gegen die Methode.

Kanban ist weit weniger stringent als Scrum und wird von Beratungshäusern, wie it-agile empfohlen, wenn noch keine Erfahrung mit agilen Methoden vorhanden ist. Für Kanban spricht, dass es viel offener als Scrum ist. Die fehlende Standardisierung der Projekte innerhalb des Unternehmens ist aber aufgrund der Überwachbarkeit nicht gewünscht.

XP eXtreme Programming ist die extremste Interpretation der Agilität. Das einzige Ergebnis das zählt ist lauffähige Software. Die Entwicklungen erfolgen in sehr kurzen Schritten, deshalb wird keine Dokumentation benötigt. Spezifikationen sind aufgrund der Kundennähe auch nicht notwendig, Unklarheiten können sofort geklärt werden. [Hruschka, Peter; Rupp, Chris; Starke, Gernot 2009] In XP wird aufgrund der Kundennähe keine Dokumentation erstellt. Dokumentation ist in den Anforderungen zwar nicht als Forderung angeführt, wird aber vom Großteil der Kunden und auch von der Geschäftsleitung verlangt. Unter der, nicht nachgewiesenen, Annahme, dass die Methode auch funktioniert, wenn Dokumentation nach Iterationen erstellt wird, kann man davon ausgehen, dass die Methode geeignet ist die Anforderungen zu erfüllen.

FDD Die sequentielle Abarbeitung der ersten drei Phasen lässt eine gewisse Starrheit annehmen, was eher der Wasserfallmethode entspricht. Durch die iterativen Entwicklungsprozesse und kurzen Zyklen kann aber auch auf veränderte Rahmenbedingungen reagiert werden. Die nachweislich, erfolgreiche Anwendung der Methode in großen, kritischen Projekten lässt es nicht zu, daraus abzuleiten, dass die Methode auch für Terna Verbesserungen bringen kann, zumal keinerlei Anwendungen im vergleichbaren Umfeld nachgewiesen werden konnten. Im Umkehrschluss kann aber auch nicht begründet werden, dass die Methode nicht geeignet wäre. Das detaillierte Studium dieser Methode ist nicht Teil dieser Arbeit.

Aufgrund der Einzelbewertungen kann man zum Entschluss kommen, dass jeder der vorgestellten agilen Methoden Aspekte vorweisen kann, die für Terna als ERP-Implementierungspartner mehr oder weniger gut geeignet sind. Keine der Methoden kann jedoch alle Anforderungen gleichermaßen abdecken. Es kann aber aufgrund dieser Bewertung nicht abgeleitet werden, dass alle agilen Methoden geeignet sind

4.4.2 Eignung für Terna

FDD wurde wegen der fehlenden Erfahrungen in vergleichbaren Projektgrößen und der sequentiellen Abbildung der Kernaktivitäten als nicht verwendbar erklärt. XP wurde wegen der extremen Kundenintegration, die organisatorisch kaum realisierbar ist, wegen der fehlenden Dokumentation und wegen der Komplexität der Methode und dem hohen Einführungsaufwand ebenfalls ausgeschieden. Für Scrum spricht die Verbreitung und Bekanntheit, dagegen spricht die fehlende direkte Kundenintegration, die Starrheit in der Anwendung der Methode und das Fehlen eines Change Request Verfahrens, das bei Werkverträgen unumgänglich ist. Gegen Kanban spricht die offene und nicht standardisierte Anwendung, die eine Überwachbarkeit und Schulung schwierig macht.

Terna ist in der Folge den Weg gegangen: "Beschränke dich nicht auf eine Methode, sondern nimm das Beste aus allen!“ [Kniberg 2010]

Die neue Methode soll schnell eingeführt werden können, weshalb aufwändige Methoden nicht in Erwägung gezogen wurden. Aber auch Methoden, die zu viele Freiheiten gewähren, wie Kanban, kommen derzeit nicht in Frage.

Über Releases und vorgelagerte Tests soll die Gesamtfunktionalität sichergestellt werden.

Es wird weiterhin Changerequests geben und zwar immer dann, wenn durch die Änderung die Kosten überschritten werden, unabhängig davon, ob das Projekt nach Aufwand, Budget oder als Fixpreis angeboten wurde. Damit Budget und Vertrag aber trotzdem eingehalten werden können, muss auf unwichtige Funktionalitäten verzichtet werden. Diese sollen durch Priorisierung am Ende herausfallen, sodass nur mehr wertschöpfende Funktionalitäten umgesetzt werden.

5 Umsetzung Agilität bei Terna

Im Nachfolgenden ist beschrieben, wie eine agile Vorgangsweise bei Terna eingeführt wurde. Terna hat sich nicht auf reine Entwicklungsvorgänge beschränkt, sondern die Agilität für gesamte Softwareimplementierungsprojekte festgelegt.

Es wird Auszugsweise aus dem [Handbuch Agile Projekte bei Terna 2016] zitiert. Die Hintergründe und Ziele stammen aus der Sicht und aus den Vorgaben der Geschäftsleitung. Der Verfasser Wolfgang Goßner hat ihm Rahmen seiner Beschäftigung mit dem Thema, wesentlich an der Festlegung der Methoden und der Erstellung des Handbuches mitgearbeitet. Die Methode verwendet Teile von Scrum, die für das Unternehmen adaptiert wurden. Aus Marketinggründen wurde aber auf Begrifflichkeiten aus Scrum so weit wie möglich verzichtet.

Die Methode heißt eagle-pm und ist ein Akronym: **e**nhanced **agile p**roject **m**ethod

Abb. 11: eagle Logo

Der Symbolik des Adlers soll Sehschärfe, Fokussierung, Agilität und Schnelligkeit beschreiben. Im Golfsport bedeutet Eagle zwei Schläge unter Par, somit besser als die Vorgabe und überperformant.

5.1 Hintergründe

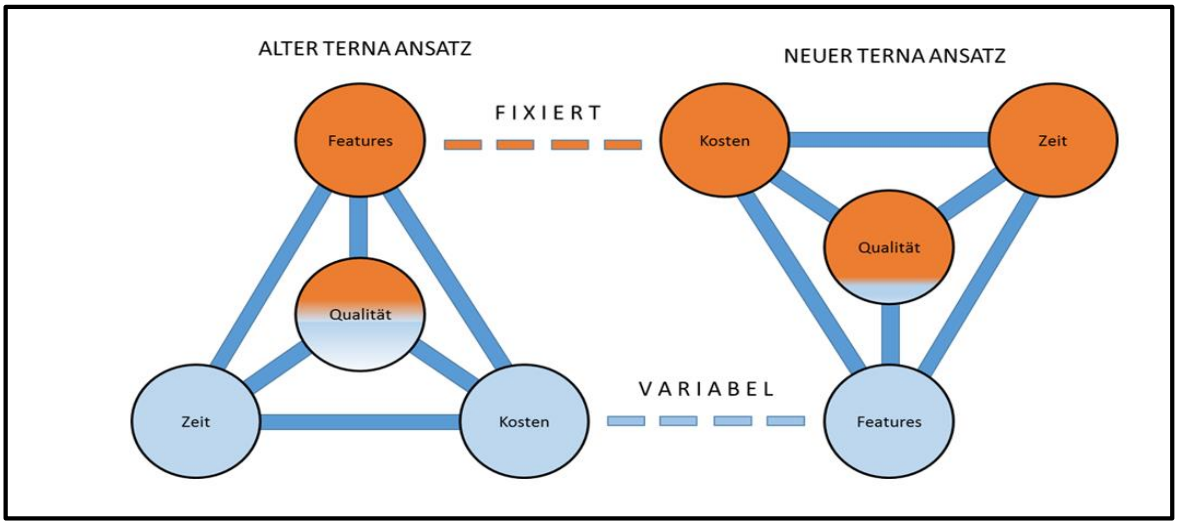
5.2 Die Ziele

Nachfolgend die Ziele der Geschäftsleitung, vertreten durch CEO Christian Kranebitter:

- Es soll kundenseitig eine bessere Verteilung des Workloads über die Laufzeit erreicht werden.
- Es muss sichergestellt werden, dass die verkaufte Lösung der implementierten Lösung entspricht.
- Durch frühe Verfügbarkeit der Lösung soll eine höhere Identifikation erreicht werden.
- Der Fokus soll von Anfang an auf eine funktionierende Gesamtlösung gerichtet sein.
- Es soll die Möglichkeit der Einflussnahme des Kunden auf die Gesamtlösung geschaffen werden.
- Durch Reduktion der Durchlaufzeiten soll das Projektrisiko deutlich reduziert werden.
- Durch interaktive, kooperative und aufbauende Vorgangsweise soll eine aktivere Beteiligung des Projektteams erreicht werden.
- Durch regelmäßige Feedbackzyklen zu funktionierenden Teillösungen oder Prototypen soll mehr Sicherheit geschaffen werden.
- Es soll mehr konkreter Output durch die Lieferung von Teillösungen oder Prototypen generiert werden.
- Durch Präsentation und Abnahme der Teillösungen und Prototypen soll mehr Ergebnisorientierung erreicht werden.

5.3 **Unterschiede**

Abb. 12: Das magische Dreieck



Die Einhaltung von Kosten und Zeit hat Vorrang, die Qualität soll gesteigert werden. Ein Maximum an Funktionalität (Features) ist nicht unbedingt das Ziel der Projekte, sondern kann auch in späteren Optimierungsprojekten realisiert werden. Welche Funktionalitäten umgesetzt werden wird durch regelmäßige Priorisierung gemeinsam mit dem Kunden festgelegt.

Das Projektteam wird bereits in der Verkaufsphase zusammengestellt und erstellt schon zu diesem Zeitpunkt ein erstes Release (Prototyp).

Es wird in Releases und Sprints, anstelle von Projektphasen geplant. Anstelle von Meilensteinen werden zu Projektbeginn ein Release– und ein Sprintplan erstellt und die Termine dafür fixiert.

In einem Release werden immer alle Geschäftsprozesse und Hauptprozesse geliefert und somit die ganze Breite der Anforderungen. Bereits beim ersten Release wird eine funktionierende Lösung oder ein Prototyp präsentiert.

Releases könnten auch produktiv gesetzt werden.

Ein Release beinhaltet vorab festgelegte Funktionalitäten und Arbeitspakete (Tasks), die im Rahmen von Sprints umgesetzt werden. Die Releases bauen aufeinander auf und nähern sich im Lauf der Zeit der Gesamtlösung immer weiter an. Unfertige Funktionalitäten werden erst im nächsten Release geliefert. Arbeitspakete werden so geschnitten, dass sie in einem Sprint umgesetzt werden können und sollten idealerweise zwischen zwei und fünf Tagen Aufwand liegen.

Im Fortschritt des Projektes erfolgt eine weitere Verfeinerung der Hauptprozesse und Detailprozesse, sodass am Ende des Projektes auch die vereinbarte Tiefe geliefert werden kann.

Sprints haben fixe Längen von vier Wochen. Releases bestehen aus einem oder mehreren Sprints.

Der Kunde bekommt schon in einer frühen Phase eine Lösung geliefert und kann damit schon testen.

Wir arbeiten in kleineren Teams, die sich voll auf das Projekt konzentrieren und enger zusammenarbeiten können.

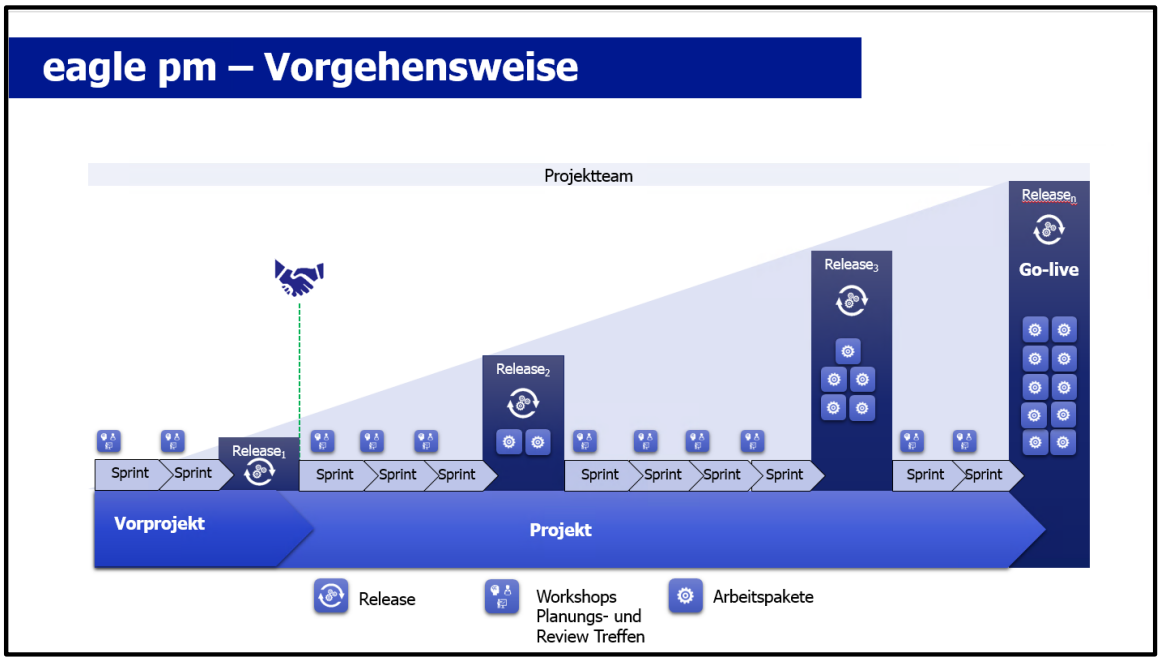
In jedem Projekt gibt es einen Lösungsverantwortlichen. Er ist dafür zuständig, dass die Lösung integriert funktioniert und erstellt gemeinsam mit dem Projektleiter den Releaseplan. Diese Rolle wird vom Berater mit der größten Erfahrung wahrgenommen.

Die Dokumentation wird auf das Notwendigste reduziert. Die inhaltliche Qualität soll wesentlich verbessert werden.

Um den Anforderungen von speziellen Branchen, wie zum Beispiel der Pharmazie gerecht zu werden, aber auch um sich nicht gleich zu Beginn als Anbieter auszuschließen, sollen auch hybride Vorgehensweisen mit Phasenkonzept und Agil möglich sein.

5.4 Das Modell

Abb. 13: Vorgangsweise eagle



5.5 Die Rollen

Kunde

Der Kunde

- ist Teil des Projektteams
- detailliert die Inhalte der Releases mit dem Projektleiter
- legt gemeinsam mit dem Projektleiter fest, welche Arbeitspakete in welchem Release realisiert werden müssen bzw. sollen
- erstellt die Arbeitspakete in Abstimmung mit dem Projektleiter auf Ebene Hauptprozess
- vereinbart den Inhalt der Sprints/Releases mit dem Projektteam
- berücksichtigt die Abhängigkeiten der Arbeitspakete untereinander
- Ist für die Integration und für die Gesamtlösung verantwortlich
- erklärt die Anforderungen im Detail
- priorisiert die Anforderungen
- testet die Umsetzungen

Projektleiter Terna

Die Rolle des Projektleiters ändert sich wesentlich und würde bei Scrumprojekten quasi der Rolle des Scrummasters entsprechen.

Der Projektleiter

- verantwortet das Team und vor allem die Zusammenarbeit im Team
- gibt die Anforderungen auf Basis des Projektvertrages vor
- erstellt einen groben Projektplan in Abstimmung mit dem Kunden
- erstellt einen Releaseplan gemeinsam mit dem Lösungsverantwortlichen
- ist budgetverantwortlich
- sitzt im Lenkungsausschuss
- führt zeitnahe Entscheidungen für den Start der Entwicklungen herbei

Lösungsverantwortlicher

Der Lösungsverantwortliche wird im Wesentlichen die Rolle des Productowners in Scrum-projekten übernehmen.

Der Lösungsverantwortliche

- detailliert die Inhalte der Releases mit dem Projektleiter und dem Kunden
- legt gemeinsam mit dem Projektleiter und dem Beraterteam fest, welche Arbeits-pakete in welchem Release realisiert werden müssen oder sollen
- erstellt die Arbeitspakete in Abstimmung mit dem Projektleiter auf Ebene Haupt-prozess
- vereinbart den Inhalt der Sprints/Releases mit dem Projektteam
- berücksichtigt die Abhängigkeiten der Arbeitspakete untereinander
- ist für die Integration und für die Gesamtlösung verantwortlich
- sitzt im Lenkungsausschuss

Berater

Der Berater

- beschreibt die Kundenanforderungen
- verfeinert und priorisiert die Anforderungen in Abstimmung mit dem Kunden
- befüllt das Backlog
- ist für das ihm zugeteilte Budget verantwortlich
- stimmt sich mit dem Kunden, den Lösungsverantwortlichen und den Entwicklern ab
- kommuniziert die Anforderungen intern
- testet Programmierungen und gibt sie intern frei
- testet Prozesse hinsichtlich der Kundenanforderungen
- konfiguriert und parametriert die Anwendung
- dokumentiert im Rahmen der Vereinbarung
- verantwortet die Funktionalität

Entwickler

Der Entwickler

- realisiert Kundenanforderungen in Abstimmung mit dem Berater
- testet gegen die Anforderung
- unterstützt bei der Dokumentation

Technisch Verantwortlicher

Der technisch Verantwortliche

› übernimmt nach der Installation durch die Systemabteilung und einer entsprechenden Übergabe die Wartung bzw. die Verantwortung über die M3 Business Engine[4]. Dazu gehören:

  ✓ die Freigabe der Installation von Standardkorrekturen des Herstellers
  ✓ die Koordination von Erstellung und Installation von Servicepacks[5]
  ✓ die Koordination der Installation von Hotfixes[6] (Kundenanpassungen, eigene Korrekturen von Standardfehler)
  ✓ Die Erteilung der Freigabe für die Installation von anderen Komponenten oder Updates

› ist zentraler Ansprechpartner für den Kunden für Angelegenheiten die M3 Business Engine betreffen und er übernimmt die Schulung für die Administration derselben.

  ✓ Dazu installiert er Spezialanwendungen und schult den Kunden in dessen Verwendung
  ✓ übernimmt die Schulung für die Übernahme von Stamm –und Bewegungsdaten und unterstützt bei der Umsetzung
  ✓ erstellt gemeinsam mit dem Kunden ein Berechtigungskonzept.

› ist zentraler Ansprechpartner für die Berater und übernimmt in diesem Rahmen:

  ✓ die Machbarkeitsprüfung bei aufwändigen Softwarekonzepten
  ✓ die Unterstützung bei der Aufwandsschätzung von Softwarekonzepten
  ✓ die Aufteilung großer Arbeitspakete auf, damit sie in einen Sprint passen und von einem oder mehreren Entwicklern bearbeitet werden können
  ✓ die Definition der Arbeitspakete bei umfangreichen Modifikationen
  ✓ die Unterstützung bei umfangreichen Modifikationen und/oder Schnittstellen, inkl. der Abstimmung und Aufnahme der Anforderung beim Kunden.

[4] M3 Business Engine, kurz M3.-Das ist die Anwendung des ERP-Systems M3 von Infor

[5] Ein Servicepack ist eine Sammlung von Programmen und Korrekturen, die gemeinsam installiert werden.

[6] Ein Hotfix ist eine Korrektur des Herstellers aufgrund meist kritischer Fehler

Projektteam

Die Teams bestehen aus dem Lösungsverantwortlichen, den Beratern und den Entwicklern, teilweise auch den Systemtechnikern und dem Technisch Verantwortlichen. Die Teams sollen klein gehalten werden, aktive Zusammenarbeit soll verstärkt werden.

5.6 **Anforderungen**

Beschreibung der Anforderungen

Eine detaillierte Erfassung der Anforderungen ist zu Beginn des Projektes nicht notwendig, sondern eingehend mit der Realisierung. Der Berater erhebt die Anforderungen gemeinsam mit dem Kunden in Form von Workshops und beschreibt sie innerhalb der Arbeitspakete. Die Anforderungen werden an das Team kommuniziert.

Auch nicht prozessrelevante Anforderungen werden so beschrieben.

Es sollte zumindest für die nächsten ein bis zwei Sprints genauere Anforderungen vorliegen.

Arbeitspakete

Arbeitspakete leiten sich aus den Detailprozessen ab und werden so verfeinert, dass sie in einem Sprint abgearbeitet werden können. Ein Arbeitspaket sollte innerhalb von maximal 5 Tagen fertiggestellt werden können. Arbeitspakete werden von dem Berater gemeinsam mit dem Keyuser des Kunden definiert. Sind die Arbeitspakete zu groß, so werden sie aufgeteilt. Bei Entwicklungstätigkeiten werden die notwendigen Tätigkeiten zwischen Berater und Entwickler in Form von Arbeitspaketen verwaltet.

Inhalt von Arbeitspaketen:

- Kundenanforderung
- Akzeptanzkriterien
- Geschätzter Aufwand
- Verantwortlichkeiten (Terna und Kunde)
- Zuordnung zu Release und Sprint
- Status (offen, bereit, in Bearbeitung, erledigt)
- Beweglich, unbeweglich
- Priorität

Product Backlog

Die Anforderungen aus den Arbeitspaketen (nicht die notwendigen Tätigkeiten) entsprechen dem Product Backlog. Das Product Backlog ist somit eine Liste aller bekannten Anforderungen funktionaler und nicht funktionaler Natur. Der Berater erstellt und verfeinert das Product Backlog.

Die Einträge sind priorisiert und geschätzt, und es sind Akzeptanzkriterien beschrieben.

Priorisierung

Es können verschiedene Methoden zur Priorisierung, wie z.B. MuSCoW oder das Kano Modell angewendet werden:

MusCow unterteilt Anforderungen in vier Gruppen:

- M - MUST (unbedingt erforderlich)
- S - SHOULD (sollte umgesetzt werden, wenn alle MUST-Anforderungen trotzdem erfüllt werden können)
- C - COULD (kann umgesetzt werden, wenn die Erfüllung von höherwertigen Anforderungen nicht beeinträchtigt wird)
- W - WON'T (wird diesmal nicht umgesetzt, aber für die Zukunft vorgemerkt

[Pichler 2008]:

5.7 **Vorgehensweise**

Projektpositionierung

Die Positionierung kann bereits in der Verkaufsphase starten und kann auch parallel zur Bearbeitung der Arbeitspakete erfolgen.

Tätigkeiten:

- Der Projektleiter legt das Projektteam fest. Idealerweise sollte das Projektteam aus den Mitarbeitern bestehen, die schon in der Verkaufsphase mitgewirkt haben.
- Auf Basis der Anforderungen oder des Vertrages werden die Geschäftsprozesse auf der Ebene von Hauptprozesse definiert.
- In Abstimmung mit dem Kunden wird ein Projektplan und ein Releaseplan erstellt und verabschiedet. Der Inhalt der Releases ist zu diesem Zeitpunkt nur grob beschrieben. Es müssen allerdings Abhängigkeiten zu anderen Projekten, sowie betriebliche Notwendigkeiten berücksichtigt werden. Wenn im Zuge von Releases auch Teilmodule in Betrieb genommen werden, so sollte das auch schon beschrieben sein.
- Gemeinsam mit dem Kunden wird festgelegt in welcher Weise und zu welchen Zeitpunkten Qualitätssicherung durchgeführt wird.
- Das Projekthandbuch wird erstellt, mit dem Kunden besprochen und verabschiedet
- Das Projektportal wird mit Microsoft Sharepoint aufgebaut.
- Projekte und Subprojekte mit den dazugehörigen Budgetverteilungen werden im internen System angelegt.
- Ein Kick-off Meeting wird durchgeführt, idealerweise gleichzeitig mit dem Beginn der Workshops.
- Das System wird seitens der Technik installiert und die Systemkonfigurationen dokumentiert.
- Die Grundschulungen werden durchgeführt. Als Ergebnis sollten die Keyuser die Bedienung des Systems verstehen und die wichtigsten Standardprozesse im System abbilden können, sowie die Grundfunktionalität verstehen. Die Schulung kann auch auf Basis eines Prototyps erfolgen.
- Es wird eine Administrator-Schulung durchgeführt.
- Berechtigungs- und Sicherheitskonzepte werden geschult und grob konzipiert.

Meilensteine:

- Das Projekt ist definiert.
- Das System ist verfügbar.
- Die Keyuser sind geschult.
- Die Administratoren sind geschult.
- Ein grober Releaseplan ist erstellt.

Release/Sprint

Tätigkeiten:

Auf Basis des groben Releaseplans erstellt der Projektleiter gemeinsam mit dem Lösungsverantwortlichen und dem Kunden einen detaillierten Releaseplan auf der Ebene von Detailprozessen. Für die nächsten ein bis zwei Releases wird auch definiert, welche Anforderungen/Arbeitspakete konkret umgesetzt werden sollen. Es ist zu berücksichtigen, dass jeweils eine funktionierende Lösung geliefert werden muss. Die Anforderungen werden vom Projektteam gemeinsam mit dem Kunden in Form von Scoping-Veranstaltungen hinsichtlich deren Priorität, Komplexität und der Beweglichkeit der Inhalte beurteilt. Unbewegliche Arbeitspakete werden gekennzeichnet und haben hohe Priorität. Ebenso werden komplexe Anforderungen und solche, die großen Kundennutzen bringen vorgereiht.

Eine Planung welche Arbeitspakete in welchen Sprints umgesetzt werden (Sprint Backlog) erfolgt vorerst intern. Dazu zählen zum Beispiel:

- Detaillierte Erhebung der Kundenanforderungen
- Ableitung des Bedarfes an Softwareerweiterungen
- Erheben der Anforderungen an die Datenmigration
- Schulung der Keyuser
- Durchführen von Funktions-, Prozess- und Integrationstests

Der Berater außerhalb von Workshops:

- konfiguriert und parametriert die Lösung
- erstellt die Entwicklungskonzepte
- testet die umgesetzten Programmierungen
- entwirft die Formulare und legt die Bericht fest
- konzipiert die Datenmigration

Der Kunde:

- beschreibt grob seine Anforderung
- erstellt die Testdokumentation
- führt Funktions-, Prozess- und Integrationstests durch
- ist zuständig für Übernahme und Test der Stamm- und Bewegungsdaten
- Erstellt die Schulungsunterlagen für die Endanwender
- Schult die Endanwender

Das Projektteam plant:

- das nächste Release und den nächsten Sprint
- die Einführung der Lösung oder Teillösung

Meilensteine:

- Getestete Teillösung oder Prototyp
- Abgenommene Teillösung oder Prototyp
- Dokumentation erstellt
- Keyuser geschult

5.8 Sprint/Releasemeeting

Im Zuge dieses Meetings werden folgende Themen besprochen:

Sprintreview
Rückblick auf den letzten Sprint

- Was war geplant?
- Was wurde fertiggestellt?
- Welche Probleme hat es gegeben?

Sprintplanung
Planung für den nächsten Sprint
Nach Analyse des Product Backlogs und Erläuterung der Anforderungen durch den Berater werden die Priorisierungen aktualisiert und gemeinsam festgelegt, welche Arbeitspakete im nächsten Sprint erledigt werden. Sind die Arbeitspakete für den Sprint zu groß, so werden sie nochmals unterteilt.
Es werden auch Abstimmungstermine, z.B. für gemeinsame Tests geplant.

5.9 Schätzen

Schätzungen basieren auf Komplexitätsvergleichen oder Aufwandsvergleichen in Relation zu anderen Anforderungen. Die Bewertung kann auch mit Storypoints erfolgen, z.B. unter Verwendung der Fibonacci Funktion [Pichler 2008]:

Tabelle 1 Fibonacci

Punkte	Semantik
0	Kein Aufwand
1	Sehr kleiner Aufwand
2	Kleiner Aufwand, doppelt so groß wie ein sehr kleiner Aufwand
3	Mittlerer Aufwand, so groß wie ein sehr kleiner und ein kleiner Aufwand zusammen
5	Großer Aufwand, so groß wie ein kleiner und ein mittlerer Aufwand zusammen
8	Sehr großer Aufwand, so groß wie ein mittlerer und ein großer Aufwand zusammen
13	Rießiger Aufwand, so groß wie ein großer und ein sehr großer Aufwand zusammen

5.10 Die Dokumentation

Allgemein

Die Dokumentation erfolgt nicht von Beginn an in detaillierter und vollständiger Form, sondern wird bei Fertigstellung der Lösung bzw. Teilfunktionalität erstellt. Teile ohne Dokumentation sind nicht abgeschlossen.

Anforderungsbeschreibung

Die Beschreibung der Anforderung erfolgt in kurzer klarer Form, kann auch in Form von Userstorys und Usecases erfolgen. Das klassische Analysedokument wird nicht mehr erstellt.

Die Beschreibung erfolgt im Projektportal innerhalb des Arbeitspaketes entweder durch direkten Eintrag oder durch Verlinkung von Dokumenten

Konfiguration

Wesentliche Konfigurationen werden durch den Entwickler oder Berater im Rahmen des Detailprozesses dokumentiert. Erfolgen Konfigurationen oder Parametrierungen durch den Kunden, so müssen sie ebenfalls dort abgelegt werden.

Solution Design

Diese Dokumente werden ebenfalls auf Ebene des Detailprozesses hinterlegt. Es wird weiterhin auch eine Liste der Gaps geführt.

Lösungsbeschreibung

Die Beschreibung der Lösung erfolgt durch den Kunden und ist gleichzeitig die Basis für die Schulungsunterlagen. Abhängig vom Projektvertrag kann sie auch vom Berater erstellt werden. Unabhängig davon, wer sie erstellt, ist sicherzustellen, dass sie erstellt wird.

Die Ablage erfolgt zusammen mit dem Arbeitspaket im Projektportal.

Testdokumentation

Die Testdokumentation erfolgt durch den Kunden.

Der Kunde bestätigt den erfolgreichen Test von Entwicklungen in der Entwicklungsaufgabe.

Workshop Protokolle

Protokolle werden sehr schlank gehalten und beinhalten keine fachlichen Themen und Beschreibungen. Aufgaben werden direkt im Projektportal erfasst. Die Erstellung passiert im Rahmen des Workshops gemeinsam mit dem Kunden.

In den Protokollen werden vermerkt:

- die anwesenden Teilnehmer
- Datum und Uhrzeiten
- die geplante Agenda
- die besprochenen Arbeitspakete
- die wichtigsten Entscheidungen

GAP-Liste

Unter GAP-Liste wird eine Kurzbeschreibung aller identifizierten Softwaremodifikationen in Listenform verstanden. Sie bezieht sich auf den jeweiligen Detailprozess oder übergreifend auf einen unterstützenden Prozess. Aus der GAP-Liste resultierende Entwicklungen werden als Arbeitspakete erfasst und auf das entsprechende GAP referenziert.

5.11 Testen

Die Akzeptanzkriterien werden im Arbeitspaket beschrieben. Das Testergebnis wird im Arbeitspaket dokumentiert. Arbeitspakete die erfolgreich getestet wurden, sind auch abgenommen.

5.12 Steuerung

Die Arbeitspakete werden in Form von Tickets erfasst. Diese Tickets werden in einem Kanban Board angezeigt.

5.13 Qualitätssicherung

Die Qualitätssicherung hängt von der Projektgröße und von der Branche ab

- Mittelständiges Projekt
  - Inhaltliche Prüfung/Begleitung durch den Lösungsverantwortlichen
  - Organisatorische Prüfung/Health-Check möglichst institutionalisiert/automatisiert, z.B. ob:
    - eine Releaseplanung vorhanden ist
    - die Projektplattform Helios aktuell ist
    - Dokumente vorhanden sind
    - AX Projekte[7] vorhanden und budgetiert sind
    - …

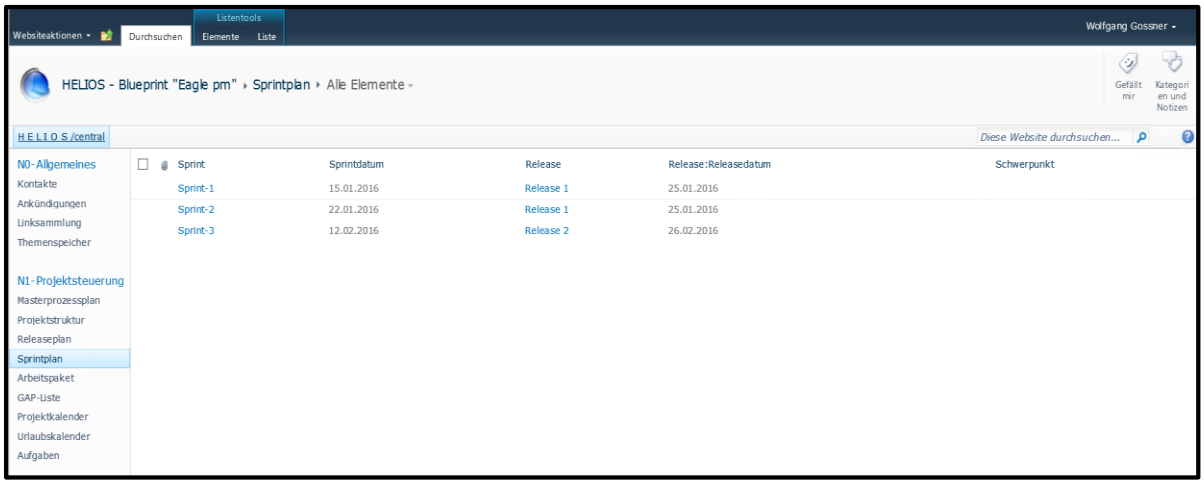
- Großprojekte (Branchenabhängig)
  - QM Prüfung durch eigenen Qualitätsbeauftragten

[7] AX Projekte sind Projekte im internen ERP-System, die unter anderem für die Zeiterfassung und Abrechnung benötigt werden.

5.14 **Die Werkzeuge**

Wesentliches Werkzeug ist die Microsoftsharepoint Plattform Helios. In dieser sind alle für das Projekt notwendigen Bibliotheken vorhanden.

Abb. 14: Helios



Zusätzlich werden noch ein Ticketsystem für die Erfassung der Arbeitspakete in der Entwicklungsabteilung und ein internes System, welches als Basis für die Budgetierung, das Controlling und die Fakturierung dient, verwendet.

6 Zusammenfassung

In diesem Kapitel werden die wichtigsten Erfahrungen aus den Projekten angeführt und die Erkenntnisse der Arbeit zusammengefasst.

6.1 Erfahrungen

Im Nachfolgenden sind Erfahrungen von Terna Mitarbeitern und Projektleitern, die basierend auf der praktizierten Methode eagle bereits Projekte umgesetzt haben oder aktuell gerade Projekte umsetzen, beschrieben.

Einführung der Methode

Durch das Fehlen externer Unterstützung bei Erarbeitung und Einführung der Methode konnten Erfahrungen aus anderen agilen Projekten nur in geringem Maße einfließen. Es wurde alles selbst mit großem Aufwand in der Theorie entwickelt und wird sich erst im Laufe der Zeit hinsichtlich der Praxistauglichkeit beweisen.

In der Anwendung der neuen Methode hat es bei den ersten Projekten mitunter auch wesentliche Fragen gegeben, die in den theoretischen Abhandlungen überhaupt nicht besprochen wurden, z.B.: Wie stellen wir sicher, dass alle Arbeitspakete als einzelne termingerecht und im Sinne des Gesamtprojektes innerhalb des Zeitrahmens fertiggestellt werden können?

Einige Kollegen fühlen sich mit der alten Methode noch sicherer und versuchen neue Projekte "klassisch" abzuwickeln. Hier gilt es noch entsprechende Überzeugungsarbeit zu leisten.

Arbeitspakete

Eine Aufteilung von Arbeitspaketen auf verschiedene Softwareentwickler war aufgrund von technischen Abhängigkeiten häufig nicht möglich. Eine Bearbeitung ein und derselben Klasse durch verschiedene Mitarbeiter ist nicht möglich. Hinzu kommt noch das Problem, dass einige Kollegen gar kein Splitten in der Realisierung von Teilfunktionalitäten wollen und dann auch nicht bereit sind, Verantwortung für die Gesamtfunktionalität zu übernehmen. In der Zusammenarbeit mit Offshore Ressourcen tun sich interkulturelle Herausforderungen auf, ebenso Probleme bedingt durch die Ferne der Kollegen und die Zeitverschiebung.

Das spätere Zusammenführen von Arbeitspaketen zu einer funktionierenden Einheit und die Überwachung der Tätigkeiten auf dieser Ebene ist derzeit noch schwierig. Das derzeitige Ticketsystem kann hier keine echte Unterstützung liefern.

Die Beschreibung der technischen Anforderungen für Offshore-Programmierer beispielsweise in Indien, muss wesentlich detaillierter erfolgen, als wenn sie für deutschsprachige Kollegen, die im Allgemeinen sogar noch im Haus sind, geschrieben werden. Hier geht man den Weg, dass man im Haus das Softwaredesign, gemeinsam mit entsprechenden Testfällen erstellt und die Codierung und den Ersttest beauftragt.

In der praktischen Anwendung waren im Blueprint Helios (Vorlage) keine Akzeptanzkriterien vorgesehen.

Es fehlen klare Definitionen wann Anforderungen bereit „ready“ sind und Arbeitspakete fertig „done“ sind. Dadurch werden die Tickets häufig hin und hergeschoben.

Die Schätzmethoden wurde nicht geschult und wurden deshalb auch nicht angewendet. Man ist es noch gewohnt jede Anforderung im Detail zu kalkulieren, auch wenn es dadurch nicht genauer, dafür aber aufwändiger wird.

Anforderungen wurden bereits wesentlich verständlicher beschrieben, als es bisher der Fall war. Hier kann man von einer deutlichen Verbesserung sprechen.

Projektmanagement

Die ersten Sprintmeetings waren unzureichend vorbereitet. Der Abschluss von Teilfunktionalitäten soll nicht erst beim Sprintmeeting erfolgen, sondern unmittelbar nach Fertigstellung dessen. Die Planung der Arbeitspakete des nächsten und übernächsten Sprints wurde aufgrund dieser Erkenntnisse bereits im Vorfeld besprochen und im Sprintmeeting, das gemeinsam mit dem Kunden stattgefunden hat, nur mehr im Detail angepasst.

Die fixe terminliche Planung der Sprintmeetings und Releasemeetings hat sich als sehr gut herausgestellt, weil immer das komplette Projektteam anwesend ist. Mit entsprechender Vorbereitung kann das Sprintmeeting in zwei bis drei Stunden abgehalten werden und die restliche Zeit für allgemeine, übergreifende Themen, Informationen und Schulungen verwendet werden. Wurden früher viele Tätigkeiten über Aufgaben gesteuert, so werden jetzt alle Tätigkeiten, die Funktionalität erzeugen in Form von Arbeitspaketen abgebildet. Das hat sich bewährt, weil es die Übersichtlichkeit und Überwachbarkeit, sowohl seitens des Kunden als auch seitens Terna wesentlich steigert.

Der Projektleitungsaufwand ist wesentlich geringer als in klassischen Projekten. Die fachliche Komponente hat der Lösungsverantwortliche übernommen. Das Team organisiert sich weitgehend selbständig. Die Planung der Ressourcen war aufgrund fehlender Aufwandsschätzungen auf Ebene der Arbeitspakete nur schwer möglich und für den Kunden nicht nachvollziehbar.

Die örtliche Trennung macht eine enge Zusammenarbeit trotz Vorhandensein von technischen Hilfsmitteln, wie Videokonferenzsysteme, Screen-Scharing, IP-Telefonie sehr schwierig. Es ist sehr schwierig mit solchen Systemen die Konzentration über mehrere Stunden aufrecht zu erhalten.

Eine Anpassung der Projektfortschrittskontrolle ist noch nicht vollständig erfolgt. Dazu gehört die Überwachung der Budgets, der Termine und der Qualität, sowie die Festlegung der notwendigen Struktur der internen Aufträge, Projekte und Subprojekte. In diesem Zusammenhang ist auch zu klären, wie detailliert die Erfassung der Tätigkeiten erfolgen muss.

Kundenakzeptanz

Es gibt in der Branche einen Trend zu Werkverträgen mit Lasten -und Pflichtenheften. Die Kunden wollen mehr Kostensicherheit und glauben es damit bekommen zu können. In den verschiedenen Fachpublikationen wird geschrieben, dass bei agilen Methoden Termin und Kosten zu Lasten der Funktionalität Vorrang haben. Damit gibt es bei vielen unserer Kunden die Befürchtung, dass zwar Termine eingehalten werden, alle Budgets verbraucht werden, aber nicht alle geforderten Funktionalitäten geliefert werden. Diese Ängste konnte man den Kunden letztendlich nicht ganz nehmen. Terna wird den Beweis antreten müssen

Kundenintegration

Die Kundenintegration ist organisatorisch schwierig, weil die Kunden im Gegensatz zu den Beratern in der Regel nicht vollständig für das Projekt abgestellt werden, sondern auch noch ihr Tagesgeschäft bewältigen müssen. Durch die permanente Beschäftigung mit der Beschreibung der Anforderungen, der Planung und Priorisierung ist die Einbindung der Kunden aber wesentlich höher als bei der klassischen Methode.

Teamgröße

Entsprechend der Theorie hat es sich auch bestätigt, dass einer der Erfolgsfaktoren kleine Teamgrößen und dadurch enge Zusammenarbeit ist. Der Abstimmungs- und Koordinationsaufwand ist wesentlich geringer, der Austausch von Erfahrung ist um ein Vielfaches gestiegen.

Dokumentation/Änderung

Vor allem bei Festpreisprojekten muss die Anforderung schon vorher klar und entsprechend detailliert sein. Hier gibt es das ständige Spannungsverhältnis zwischen fixen Kosten, Terminen und änderbaren Anforderungen, das so logischerweise nicht gelöst werden kann. Wie schon ausgeführt gibt es nur die Möglichkeit auf Funktionalität zu verzichten oder eben Vertragsmodelle wie den agilen Festpreis [It-Agile 2013] zu verwenden.

6.2 Eignung

6.2.1 Eignung für Terna

Aufgrund der theoretischen Bewertung (siehe 4.4), der praktischen Erfahrungen (siehe 6.1) kann man davon ausgehen, dass agile Methoden für Terna geeignet sind die Anforderungen zu erfüllen und Verbesserungen in den Projekten zu bringen. Keine der Methoden kann aber alle Wünsche gleichermaßen erfüllen, sodass eine an das Unternehmen angepasste Methode, die sich die wichtigen Teile aus den bekannten Methoden nimmt und entsprechend kombiniert, ideal erscheint. Auch in der Literatur sind viele Empfehlungen zu finden dies so zu tun. Kniberg [Kniberg 2010]. Boehm und Turner [Boehm und Turner 2004] empfehlen in gewissen Fällen sogar eine Mischung klassischer und agiler Methoden, was laut einer Studie der Hochschule Koblenz [Komus 2014] auch der häufigste Fall in der Praxis ist.

Die Aussage der Eignung muss aber insofern relativiert werden, als diese nur für das jetzige von Terna betreuten Geschäftsfeld gilt. Für Projekte in anderen Ländern, für andere Branchen und Unternehmensgrößen oder mit anderer Software liegen an dieser Stelle keine Erfahrungen und Anforderungen vor. Somit muss die Eignung noch unter den Aspekten der Branche, der Software, der Projektgröße und der Länder betrachtet werden.

Zu den von Terna mit ERP-Systemen bedienten **Branchen** gehören: produzierende Unternehmen, Handel mit Produkten und Dienstleistungen, Serviceunternehmen und Vermieter von mobilen Investitionsgütern. Für diese kann angenommen werden, dass agile Methoden geeignet sind. Unternehmen, die klassische Vorgehensweisen für die Softwareentwicklung vorschreiben, wie Militär, Pharma und öffentliche Behörden sind für reine agile Vorgehensweisen nicht geeignet.

Bei den Prinzipien agiler Methoden [Agile Manifesto 2001] geht es an keiner Stelle um die Frage um welche **Software** es sich handelt. Die große Verbreitung agiler Methoden in der Softwareentwicklung [Komus 2014] lässt es zu, anzunehmen, dass es nicht relevant ist, ob es um ERP-Software oder andere Art von Software geht. Auch lassen sich dazu keine entsprechenden Hinweise in der Literatur dazu finden. Im Detail können die Methoden aber abhängig von der Art der Software mehr oder weniger geeignet sein. Kanban wird in vielen Unternehmen ohnehin auch außerhalb der IT für Projekte aller Art verwendet.

Der Einfluss der **Projektgröße** wurde in dieser Arbeit nicht analysiert. Es sind aber nicht alle Methoden in gleicher Weise für große Projekte geeignet. Pichler [Pichler 2008] spricht im Falle von Scrum von großen Projekten, wenn es mehr als ein Team gibt und bei verteilten Projekten, wenn die Teams an mehr als einen Ort arbeiten und stellt auch entsprechende Methoden zur Skalierung und Verteilung vor. FDD kann nachweislich Erfolge bei großen Projekten nachweisen [De Luca 1998]. Bei Kanban gibt es eigentlich keine Beschränkung der Teamgröße. Bei XP wird die ideale Teamgröße mit 12 bis 14 Personen

begrenzt [Bunse und Knethen 2008]. Abhilfe schaffen können hier sogenannte Metateams, also ein Team aus Teams [Bunse und Knethen 2008].

Ob Projekte oder Kunden in Ländern, die von Terna heute nicht betreut werden für agile Methoden mehr oder weniger geeignet sind, lässt sich an dieser Stelle nicht beantworten.

Agile Methoden für die Softwareentwicklung bei Terna als ERP-Implementierungspartner sind somit geeignet, solange vom Kunden nicht klassische Vorgehensweisen vorgeschrieben werden. Eine Eignung für Projekte in anderen, von Terna nicht betreuten Ländern kann nicht abgeleitet werden.

Es stellt sich nun die Frage, ob man auf Basis dieser Erkenntnisse die Eignung und Verbesserung für ERP-Implementierungspartner im Allgemeinen ableiten kann.

Unter der Annahme, dass andere ERP-Implementierungspartner, die zu Terna im Wettbewerb stehen auch ähnliche Anforderungen haben, kann man durchaus annehmen, dass die Eignung in gleicher oder ähnlicher Weise gegeben ist.

Zusammenfassend kann man sagen, dass agile Methoden für die Softwareentwicklung bei ERP-Implementierungspartnern geeignet sind. Anpassungen an die Besonderheiten des Unternehmens, aber auch an die Kunden und daraus abgeleitet ein Mix aus den Teilen der Methoden, die das beste Ergebnis bringen können, sind die Voraussetzung für erfolgreiche Projekte. Ebenso müssen die Voraussetzungen im eigenen Haus geschaffen werden, die mitunter auch nicht kulturelle Änderungen notwendig machen, welche kurzfristig nicht umgesetzt werden können. Die Besonderheiten der Projekte und Anforderungen der Kunden dürfen ebenfalls nicht außer Acht gelassen werden. Der Kunde gibt letztlich auch die Rahmenbedingungen vor, nach denen sich die Partner richten müssen.

7 Literatur

[Agile Manifesto 2001] http://agilemanifesto.org/iso/de/, veröffentlicht, am 04.06.2016, um 15:30

[Beck 2004] Beck, Kent: Extreme Programming Explained-Embrace Change, Addison-Wesley, 2. Auflage, 2004

[Boehm und Turner 2004] Boehm, Barry W. Turner, Richard: Balancing agility and discipline, Boston, Addison-Wesley, 2004

[Bunse und Knethen 2008] Bunse, Christian; Knethen, Antje von: Vorgehensmodelle kompakt. 2. Aufl. Heidelberg: Spektrum Akad. Verl. (Kompakt-Reihe, 2008

[De Luca 1998] Jeff de Luca, http://www.nebulon.com/articles/fdd/latestprocesses.html, veröffentlicht am 20.02.1998

[Dräther Koschek Sahling 2013] Dräther, Rolf; Koschek, Holger; Sahling, Carsten: Scrum-Kurz und gut, O`Reilly Verlag,2013

[Goldratt 1999] Goldratt Eliyahu: Theory of constraints-What is this thing called and how should it be implemented?, North River Press, Croton-on-Hudson, NY, 1999

[Handbuch Agile Projekte bei Terna 2016] Wolfgang Goßner, Simone Sieberer: Handbuch Agile Projekte bei Terna, Version 03, Terna, 2016

[Hruschka, Peter, Rupp, Chris; Starke, Gernot 2009] Hruschka, Peter; Rupp, Chris; Starke, Gernot: Agility kompakt, Tipps für erfolgreiche Systementwicklung
Spektrum Akad. Verl; Springer, 2. Auflage, 2009

[It-Agile 2013] http://www.it-agile.de/verschiedene-festpreismodelle.html, veröffentlicht am 01.04.2013 15:57:48

[Kniberg 2010] Kniberg, Henrik; Skarin, Mattias: Kanban and Scrum-Making the most of both, C4Media, Inc., 2010

[Komus 2014] Status Quo Agile 2014, zweite Studie des BPM-Labors der Hochschule Koblenz, Prof. Dr. Ayelt Komus, über die Verwendung agiler Methoden

[Leopold und Kaltenecker 2012] Leopold, Klaus; Kaltenecker: Siegfried-Kanban in der IT-Eine Kultur der kontinuierlichen Verbesserung schaffen, Hanser Verlag, München, 2012

[Ludewig und Lichter 2010] Jochen Ludewig, Horst Lichter: Softwareengineering Grundlagen, Menschen, Prozesse, Techniken, 2. Auflage, dpunktverlag,2009

[Neus, Trompeter, Mandischer 2011] Neus, Sebastian; Trompeter, Jens; Mandischer, Martin: Scrum Kompakt, itemis AG, Lünen. 2011

[Pichler 2008] Pichler, Roman: Srum - agiles Projektmanagement erfolgreich einsetzen, Dpunkt-Verlag, Heidelberg, 2008

[Prowell Trammell Linger Poore 1999] Stacy J. Prowell, Carmen J. Trammell, Richard C. Linger, Jesse H. Poore: Cleanroom. Software Engineering-Technology and Process, Addison-Wesley, 1999

[Rational 2016] Rational Unified Process Best Practices for Software Development Teams-Rational Software White Paper TP026B, Rev 11/01

[Rupp 2014] Rupp, Chris: Requirements-Engineering und -Management- Aus der Praxis von klassisch bis agil, Hanser Verlag, München, 2014

[Sommerville 2011] Sommerville, Ian: -Software engineering 9th edition, Pearson, Boston, 2011

[Wikipedia 2016-1] https://de.wikipedia.org/wiki/Hidden_Markov_Model, veröffentlicht, am 15.01.2016 um 08:30 Uhr

[Wirtschaftslexikon 2016-1] http://wirtschaftslexikon.gabler.de/Archiv/17984/enterprise-resource-planning-system-v12.html, verfügbar am 30.03.2016, 08:15 Uhr

[Wirtschaftslexikon 2016-2] http://wirtschaftslexikon.gabler.de/Archiv/17984/enterprise-resource-planning-system-v12.html#head2, verfügbar am 30.08.2015

[Wolf, Solingen, Rustenburg 2011] Wolf Henning von, Soninge, Rini von, Rustenburg, Elvo: Die Kraft von Scrum, Inspiration zur revolutionärsten Projektmanagement-Methode, München, Addison-Wesley, 2011

8 Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Jenbach, den 30. Juni 2016

Wolfgang Goßner